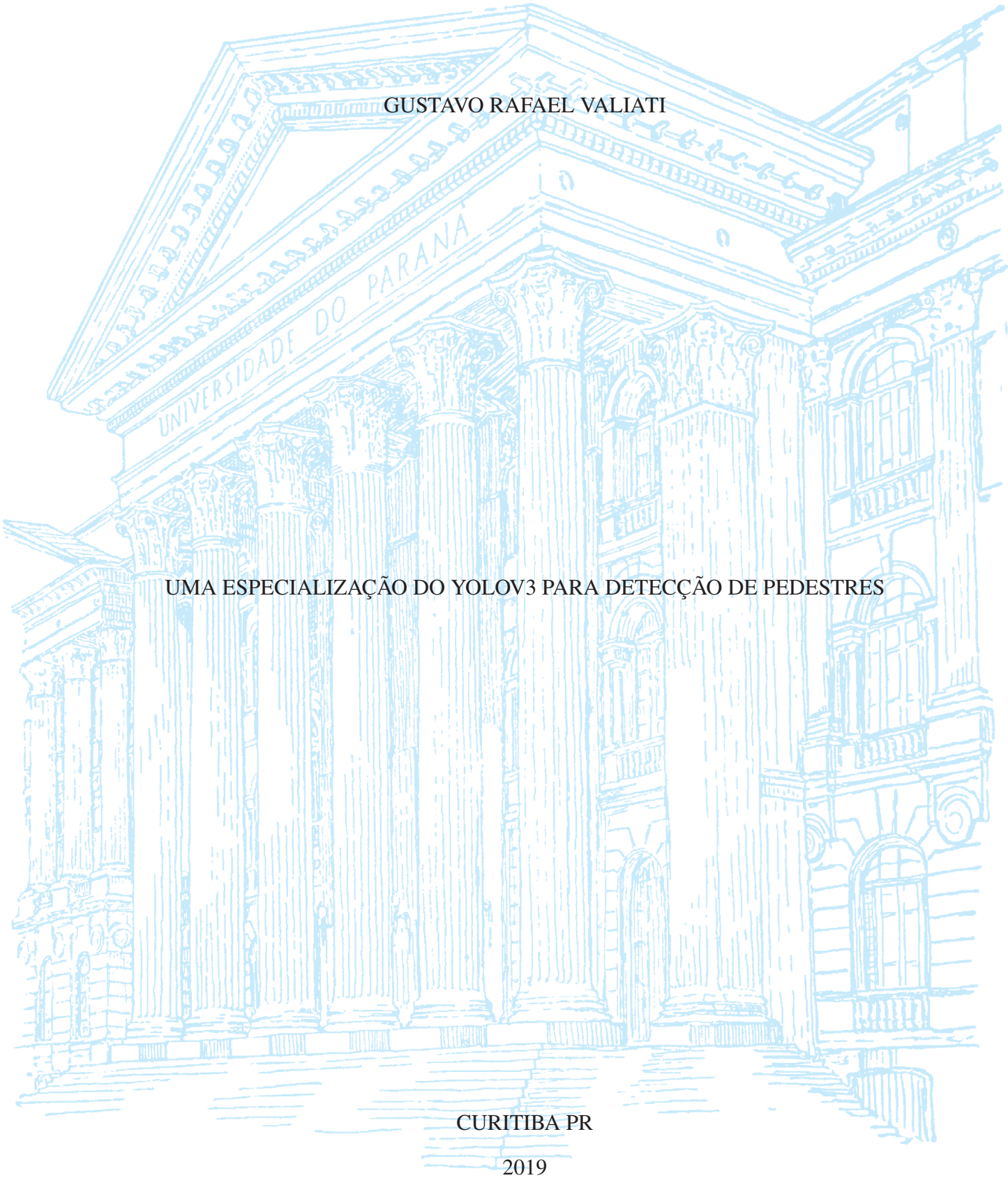
UNIVERSIDADE FEDERAL DO PARANÁ

GUSTAVO RAFAEL VALIATI



UMA ESPECIALIZAÇÃO DO YOLOV3 PARA DETECÇÃO DE PEDESTRES

CURITIBA PR

2019

GUSTAVO RAFAEL VALIATI

UMA ESPECIALIZAÇÃO DO YOLOV3 PARA DETECÇÃO DE PEDESTRES

Dissertação apresentada ao curso de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Mestre em Informática..

Orientador: David Menotti Gomes.

CURITIBA PR

2019

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

V172e Valiati, Gustavo Rafael
Uma especialização do YOLOV3 para detecção de pedestres [Recurso eletrônico] / Gustavo Rafael Valiati – Curitiba, 2019

Dissertação (mestrado) - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-graduação em Informática.
Orientador: David Menotti Gomes

1. YOLOv3 (detector genérico de objetos de tempo-real). 2. Videomonitoramento. I. Universidade Federal do Paraná. II. Gomes, David Menotti. III. Título.

CDD: 006.2

Bibliotecária: Roseny Rivelini Morciani CRB-9/1585

MINISTÉRIO DA EDUCAÇÃO
SETOR SETOR DE CIENCIAS EXATAS
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO INFORMÁTICA - 40001016034P5

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **GUSTAVO RAFAEL VALIATI** intitulada: **Uma Especialização do YOLOv3 para Detecção de Pedestres**, após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 25 de Fevereiro de 2019.

DAVID MENOTTI GOMES
Presidente da Banca Examinadora (UFPR)

p/

WILLIAM ROBSON SCHWARTZ
Avaliador Externo (UFMG)

DANIEL WEINGAERTNER
Avaliador Interno (UFPR)

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
Setor de Ciências Exatas
Coordenação de Pós-Graduação em Informática

A Deus que me oferece o Caminho, a Verdade e a Vida. A minha família que me apoia. Aos professores que me instruem. Aos meus amigos que me incentivam.

AGRADECIMENTOS

Agradeço a Deus pela oportunidade de iniciar e concluir este desafio, e que me abençoou o tempo todo, sendo gracioso e me concedendo força.

Agradeço a Universidade Federal do Paraná pela viabilização e estruturação deste curso.

Agradeço aos professores que fizeram parte da minha formação, especialmente ao meu orientador, o professor David Menotti Gomes, que decidiu investir em mim, me suportou, instruiu e lutou comigo até o final. Agradeço também aos professores que aceitaram fazer parte das minhas bancas de defesa: William Robson Schwartz, Daniel Weingaertner e Luiz Eduardo S. Oliveira.

Agradeço aos colegas do Laboratório de Visão, Robótica e Imagem, que me auxiliaram no dia a dia dos trabalhos, especialmente a Cides Bezerra e Rayson Laroca.

Agradeço aos colegas do PPTIC e CELTAB que me apoiaram e incentivaram neste desafio.

Agradeço a Fundação Parque Tecnológico Itaipu - Brasil pelo apoio na execução das atividades relacionadas ao Mestrado, e também a liberação de uma das bases de dados deste projeto.

Agradeço aos meus pais, Jaqueline e Ivo Valiati, por todo o apoio concedido em busca do meu sucesso, em que se sacrificaram inúmeras vezes para que eu pudesse usufruir desta oportunidade, me suportaram em dias difíceis, me incentivaram a não desistir, e acreditaram desde sempre em mim.

Agradeço aos meus tios Maria e Mario Valiati, e meu primo Rafael Valiati, que me receberam carinhosamente em Curitiba e me apoiaram nas minhas idas e vindas em prol da superação do desafio.

Agradeço aos meus amigos que acreditaram no meu sucesso, e me apoiaram bastante, em especial: Thiago e Elyn Medeiros, Fabio Barboza, Jônios Máximo e Aiman Amim.

Agradeço a todos que, direta ou indiretamente, fizeram parte da minha formação.

RESUMO

A Detecção de Pedestres é uma tarefa da Visão Computacional que trabalha na localização de pedestres em imagens/vídeos para aplicações como assistência de direção, videomonitoramento, interfaces humanas, veículos e robôs autônomos. Progressos nestas aplicações podem se refletir na melhoria da qualidade de vida, e por isso, elas vem recebendo considerável atenção nos últimos anos. Na área de Aprendizagem de Máquina, Redes Neuronais Convolucionais Profundas têm sido utilizadas como principal ferramenta na obtenção dos melhores resultados em diversos desafios de detecção. Apesar do contínuo progresso na tarefa, ela ainda não está saturada, e há espaço para melhorias, inclusive para atingir-se o nível da acurácia humana. Há uma tendência entre os métodos de detecção em que tipicamente procuram aumentar a acurácia através do uso de modelos cada vez mais complexos, que elevam os custos computacionais, normalmente comprometendo a velocidade de detecção. A velocidade de detecção tem se revelado tão importante quanto a acurácia, monstrando impactar diretamente em tarefas como monitoramento, segurança automotiva e robótica. Neste trabalho, esta tendência é contrariada. Em uma primeira abordagem, o detector genérico de objetos de tempo-real, YOLOv3, é levado para experimentação no desafio Caltech Pedestrian Detection Benchmark, para avaliação de sua acurácia e velocidade de detecção contra os melhores trabalhos do desafio. Para conseguir isso, o YOLOv3 é movido de um domínio multiclasse (por exemplo, COCO Dataset com 80 classes) para a tarefa específica de detectar uma única classe, isto é, pedestres. Foi possível demonstrar que o YOLOv3 é mais rápido que os três melhores trabalhos do desafio, e ao mesmo tempo possui acurácia consistente. Em uma segunda abordagem, a técnica de "infusão de segmentação semântica fraca" é utilizada para modificar a rede neural do YOLOv3. Desta forma, o método apresentou uma detecção de pedestres aprimorada, sem impacto na velocidade de detecção, colocando o YOLOv3 na décima segunda posição do desafio Caltech, ficando apenas 2,94% atrás do melhor método da métrica principal. Adicionalmente, uma nova base de dados de detecção de pedestres é introduzida, sendo baseada no circuito de videomonitoramento do Parque Tecnológico Itaipu. Quase 8.000 frames compõe o dataset, oriundos de 21 câmeras, contendo mais de 30.000 pedestres divididos em 8 classes.

Palavras-chave: Detecção de Pedestres, Videomonitoramento, YOLO, Caltech Pedestrian Dataset, PTI01 Pedestrian Dataset

ABSTRACT

The Pedestrian Detection is a Computer Vision task which works on locating pedestrians in images/videos for applications like driving assistance, video surveillance, human interfaces, autonomous vehicles, and robots. Progresses on those applications are likely to enhance the quality of life, and because of that, they have been receiving considerable attention in the last years. In the Machine Learning area, Deep Convolutional Neural Networks (DCNN) have been the main tool in achieving the best results in many detection challenges. Despite the continuous progress in the task, it is not saturated yet, and there is room for improvements, even to reach the human-accuracy level. There is a common tendency between the detection methods to increase the accuracy typically by making use of every time more complex models which elevate the computational costs, normally compromising the detection speed. The detection speed has shown to be as important as the accuracy, demonstrating to have a direct impact on tasks like surveillance, automotive safety, and robotics. In this work, we go in the opposite direction of the trend. In our first approach, we bring the YOLOv3, a real-time generic object detector, for experimentation in the Caltech Pedestrian Detection Benchmark, in order to evaluate its accuracy and speed against the top works in such a challenge. To accomplish that, YOLOv3 is moved from a multiclass domain (e.g., COCO Dataset with 80 classes), to the specific task of detecting a single class, that is, pedestrians. We have demonstrated that it is faster than the top three works while having consistent accuracy. In a second approach, we propose to use the "weak semantic segmentation infusion" technique by modifying the YOLOv3's network. The method demonstrated to enhance the pedestrian detection with no impact on the detection speed, placing the YOLOv3 in the 12th position in the Caltech Benchmark, staying 2.94% behind the best method in the main metric. Additionally, we introduce a pedestrian detection dataset based on the Itaipu Technological Park's video surveillance system. Almost 8,000 thousand frames compose the dataset from 21 cameras and more than 30,000 pedestrians spread in 8 classes.

Keywords: Pedestrian Detection, Video Surveillance, YOLO, Caltech Pedestrian Dataset, PTI01 Pedestrian Dataset

LIST OF FIGURES

LIST OF TABLES

List of Acronyms

ACF	Aggregated Channel Features
AGMM	Adaptive Gaussian Mixture Model
AI	Artificial Intelligence
ANN	Artificial Neural Network
AP	Average Precision
API	Application Programming Interface
BCN	Binary Classification Network
CNN	Convolutional Neural Network
Conv-LSTM	Convolutional Long-short Term Memory
DCNN	Deep Convolutional Neural Network
DINF	Departamento de Informática
DPM	Deformable Part Model
FCN	Fully Convolutional Neural Network
FN	False Negative
FP	False Positive
FPPI	False Positives per-image
FPS	Frames per Second
GPU	Graphics Processing Unit
GR	General Recall
GT	Ground-Truth
HOG	Histograms of Oriented Gradient
ICF	Integral Channel Features
IoU	Intersection-over-Union
LAMR	Log-Average Miss Rate
LSTM	Long-short Term Memory
mAP	mean Average Precision
MOTChallenge	Multiple Object Tracking Benchmark
ms	milliseconds
NMS	Non-maximum Suppression
PPGINF	Programa de Pós-Graduação em Informática
PTI	Itaipu Technological Park
PTI01	PTI01 Pedestrian Dataset
ReLU	Rectified Linear Unit
RN	Real Negative
RNN	Recurrent Neural Network
RP	Real Positive
RPN	Region Proposal Network
SSD	Single Shot Multibox Detector
SVM	Support Vector Machine
TLL	Somatic Topological Line Localization

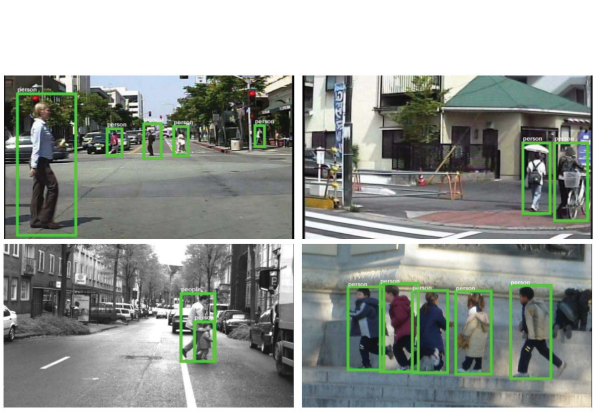
TN	True Negative
TP	True Positive
UFPR	Universidade Federal do Paraná
VRI	Visão, Robótica e Imagens
wAP	weighted mean Average Precision

CONTENTS

1 INTRODUCTION

Among the many practical application areas supported by computational tasks, there is a variety of such activities that have great potential to positively impact the quality of life (Dollár et al., 2012). Those applications are related to robotics, biometry, driving assistance, traffic law enforcement, intelligent video surveillance, autonomous vehicles, assistive technology, augmented reality, advanced human interfaces, transit safety and others (Dollár et al., 2009b; Mao et al., 2017; Zhang et al., 2017; Correia, 2016; Huang and Ramanan, 2017; Duan et al., 2017; Correia, 2016; Liu et al., 2018; Brazil et al., 2017; Laroca et al., 2018).

The **Pedestrian Detection** ended up being a crucial computational task for the execution of some of such applications (Guo et al., 2017; Dollár et al., 2012). In Artificial Intelligence (AI), Pedestrian Detection corresponds to the action of locating pedestrians in images or videos (Xing et al., 2017) (Figure 1.1(a)). According to Zhang et al. (2017), because the relevance, it became a popular topic in the Computer Vision (subarea of the AI), and in the last decade it has received active research efforts mostly because the direct applications in robotics and surveillance (Correia, 2016; Brazil et al., 2017). For Wang et al. (2018), Pedestrian Detection is one of the most challenging problems in Computer Vision.



(a) Pedestrian Detection. Source: (Dollár et al., 2012).

(b) Object Detection. Source: (Ren et al., 2015).

Figure 1.1: Examples of Object and Pedestrian Detections.

For Dollár et al. (2012), detecting and tracking people are relevant for research because they are important components of a machine's environment. This would help machines in interacting with humans, being one of the most potentially useful challenges for modern engineering. For instance, the automatic detection of pedestrians applied to ground vehicles would support in reducing the number of pedestrian injuries and fatalities. For Brazil et al. (2017), Pedestrian Detection has a significant impact in safety for urban autonomous driving.

Molchanov et al. (2017) give another example. They affirm that more than 80% of the monitoring done by video surveillance systems is focusing on people. Those systems play an essential role in public security (Zhang et al., 2016a; Duan et al., 2017; Dollár et al., 2009b), and their demand is likely to have increased due to the growing need for public safety and the popularization of surveillance video networks (Ye et al., 2016; Zheng et al., 2016a). Pedestrian Detection has many uses in such systems, being one of the critical components for danger understanding, risky behavior, violence detection, weapon detection, and person re-identification,

for example (Zhang et al., 2016a; Huang and Ramanan, 2017; Ko and Sim, 2018; Guo et al., 2017; Lai and Maples, 2017; Zheng et al., 2016a; Gajjar et al., 2017). According to Zheng et al. (2016a), these tasks depend on the automation of pedestrian detection which is a subtask of such activities, mainly because it is unreasonable to use human labor to execute such detection action as a component of those systems.

Despite being an independent task, Pedestrian Detection is a specialized application of the Generic Object Detection, which the latter refers to locating objects from a variety of categories, not only *pedestrians* but cars, dogs, bicycles or any other object (Figure 1.1(b)). Brazil et al. (2017) declare that among the studied problems in object detection, Pedestrian Detection is one of the most studied ones due to its real-world importance.

The object detection task is one of the most fundamental and challenging problems in Computer Vision (Liu et al., 2018). Specifically for Pedestrian Detection, there are many challenges in locating people in images such as appearance and clothing variation, a wide range of poses due to the high articulation of human body, different illumination between scenes, occlusions, scale, low resolution and high computational processing requirements (Correia, 2016; Dalal and Triggs, 2005; Brazil et al., 2017; Varga and Szirányi, 2017).

According to Liu et al. (2018), object detectors pursuit two main goals: high accuracy and high efficiency. While accuracy refers to the quality of the detections by correctly locating the objects in the scene, the efficiency is about doing the job using the smallest quantity of computational resources possible, that is, processing, memory, storage, and so on.

Pedestrian Detection has been progressively evolving since the past decade. The primary efforts tackle the detection accuracy by feature enhancements, and speed by working on computational costs (Dollár et al., 2010; Correia, 2016). Even with continuous improvements in the last years due to the constant effort in solving the Pedestrian Detection problem, there is no sight for saturation on this task yet. Recent works still show consistent progress, and there is much terrain until the state-of-the-art reaches human performance (Zhang et al., 2016b). Liu et al. (2018) state that object detection still has an expressive room for improvements before satisfying the needs of practical applications.

1.1 PROBLEM

Lately, the increase of accuracy in Pedestrian Detection has been proportional to the increase in computational costs (Dollár et al., 2010). Liu et al. (2018) state that despite significant progress in the detection tasks in the last years, the works do not present a solution that satisfies both goals of accuracy and speed. The task has been tackled with complex solutions which use heavy computational methods (Guo et al., 2017).

Liu et al. (2018) conclude that the efficiency of Convolutional Neural Network (CNN) features is a object detection domain which still requires some investments. The current most successful pedestrian and object detection methods, according to two well-known challenges[1], are detectors based on Deep Convolutional Neural Networks (DCNNs) (Brazil et al., 2017; Song et al., 2018; He et al., 2017; Redmon and Farhadi, 2018; Lin et al., 2017), which are computationally heavy, but accurate. There is a tendency of increasing the depth of DCNNs in order to increase its accuracy. The deeper a CNN is, the more parameters are required to be calculated, more training data is necessary, and more powerful hardware become essential. These circumstances do limit the real-time and embedding potential of the detectors. In other words, detection speed is compromised and depends on hardware with high processing performance (Correia, 2016; Liu et al., 2018).

[1]Caltech Pedestrian Detection Benchmark and COCO Object Detection Task

According to Liu et al. (2018), besides the efficiency of CNNs, there is another domain that also should receive more attention: the research of more efficient detection frameworks. The most efficacious detectors are region-based (two-stages) or based on unified pipelines (one-stage). The first usually presents the best accuracies while being the most computational intensive ones. The latter typically does not present the higher accuracy but are less computationally expensive.

To Dollár et al. (2010), the speed of Pedestrian Detection is relevant, and it is fundamental for applications including robotics, human-machine interfaces, automotive safety, and surveillance. Varga and Szirányi (2017) also state that video surveillance detection speed is as important as accuracy. According to Liu et al. (2018), there is another reason for the need of scalable and efficient detectors: the demanding analysis of visual data in an environment of an increasing number of images from social media networks and mobile devices.

1.2 MOTIVATION

We have seen that the literature considers pedestrian detection important for many reasons, such as autonomous driving or assistance, automation of monotonous tasks enabling humans to focus on more important/intelligent activities, support of public security through intelligent video surveillance, assistive technology for disabled and elderly, among others. It is also clear that this kind of detection is far from being a solved task in Artificial Intelligence. Despite the continuous progress achieved in the last years, there is much to improve yet. It was possible to observe that the improvement in detection quality has been powered by the increase in neural network complexity and, by consequence, processing requirements. While it is common for detectors sacrificing speed to obtain more accuracy, it is notorious that many real-world applications may benefit from fast detection, such as autonomous driving or video surveillance (Du et al., 2017).

With the presented context, we understand that for object detection there is a significant, available and still not solved field of research: the trade-off between detection accuracy and speed (Du et al., 2017). We also note that this field, consequently, is unsolved for pedestrian detection as well. It is possible to confirm that situation when analyzing the state-of-the-art works in renowned detection challenges in the literature, such as *Microsoft COCO Dataset* (COCO) (Lin et al., 2014) and *Caltech Pedestrian Dataset* (Caltech) (Dollár et al., 2009b). In both dataset benchmarks, the top accuracy detectors are not necessarily also the fastest ones.

When interpreting the results from both benchmarks, we see an opportunity of exploration. YOLOv3 (Redmon and Farhadi, 2018) is the fastest object detector on COCO, being 4× faster than the most accurate one, while also having top-tier detection quality. Simultaneously it is possible to verify in the literature that detection speed has its importance for practical and real-world applications, including automotive related ones. The Caltech Pedestrian Dataset was created in the context of automotive applications, built with data collected from a moving vehicle in an urban environment. Consequently, it is expected that accurate and fast detectors would be submitted to the Caltech's evaluation. Curiously, to the best of our knowledge, the YOLOv3 has not been reported in the official Caltech pedestrian detection benchmark yet, and there is no evidence of such evaluation in the literature. Because of that, we understand as relevant the analysis of accuracy and speed performance of YOLOv3 on Caltech. Being fast and accurate, YOLOv3 could demonstrate significant results for pedestrian detection in this well-known challenge. We put this analysis as the **first contribution** of our work.

Additionally, due to the importance demonstrated in the literature on pedestrian detection in **video surveillance** scenarios, it is desired to evaluate the performance of YOLOv3 on a real-world dataset for such environment. That is our **second contribution**. Despite the existence of a few datasets of that kind, none of them represented a surveillance scenario similar to the one

we have access at the Itaipu Technological Park (PTI)[2]. The dataset was built from the PTI's video surveillance system that has more than 250 cameras spread over the park, that monitors up to 6 thousand people per day. Inspired by that scenario, we decided to build a realistic video surveillance dataset composed of images from 21 cameras, almost 8,000 images and more than 28,000 high quality annotated pedestrians. The dataset has been named *PTI01 Pedestrian Dataset* and is available under PTI terms and request.

Our **final contribution** is a report of an experimental modification in the YOLOv3 model for the pedestrian detection scenario. We apply a technique that could potentially increase its accuracy performance without impact in the detection speed. We have seen in the literature that it is typical for detectors to increase its detection quality by increasing the complexity in network compositions like the use of deeper CNNs, which up-escalates computational costs. Contrary to that, we see that detection speed has been considered relevant. Among the state-of-the-art works, we find one technique that does not follow the tendency of increasing processing demands to achieve higher accuracy. It is called *weak semantic segmentation infusion*, and have been used by Brazil et al. (2017) to improve the model's ability in detecting pedestrians while not affecting the inference time. We propose to modify the YOLOv3 architecture to include this technique, trying to improve its pedestrian detection quality without impacting the model's speed.

1.3 OBJECTIVES

This work has as primary objective the analysis of the YOLOv3 (Redmon and Farhadi, 2018) as a pedestrian detector through evaluations in the Caltech Pedestrian Detection Benchmark, as well in the PTI01 Pedestrian Dataset. We aim to understand the cost-efficiency of the detection quality and the detection speed compared to other state-of-the-art works, additionally trying to improve the YOLOv3's pedestrian detection ability by using an infusion technique.

Deriving from the primary objective, the following specific ones are defined:

- Build up the PTI01 Pedestrian Dataset by the selection and annotation of around 8,000 images from the PTI's video surveillance network;
- Train and evaluate the YOLOv3 model with different configurations on Caltech and PTI01 datasets;
- Compare the YOLOv3 performance to the top works on the Caltech Pedestrian Detection Benchmark;
- Modify, train and evaluate the modified version of the YOLOv3's network architecture by the application of the "weak semantic segmentation" infusion technique utilized by the SDS-RCNN (Brazil et al., 2017) method;
- Compare the YOLOv3 performance with and without technique in both PTI01 and Caltech datasets;

1.4 CONTRIBUTIONS

Following the objectives established, this work presents the resulting contributions:

- Comparative study of the YOLOv3's performance on Caltech and PTI01 datasets using different parameters and configurations;

[2]www.pti.org.br

- Analysis of the effect *weak semantic segmentation infusion* technique coupled to the default YOLOv3 architecture both in PTI01 and Caltech datasets;
- Release of the best trained models at the project's repository[3];
- Release of the PTI01 Pedestrian Dataset, under PTI's terms and requirement process;
- Release of the modified source code for the YOLOv3 architecture in the Keras framework at https://github.com/gustavovaliati/keras-yolo3, including the infusion technique and dozen of the related algorithms;

1.4.1 Papers

1. Valiati, G.R., Menotti, D. (2018). A Preliminary Evaluation of Pedestrian Detection on Real-World Video Surveillance. In the 22nd International Conference on Image Processing, Computer Vision, & Pattern Recognition (IPCV 2018). Qualis CC: B5. Status: Published;
2. Valiati, G.R., Menotti, D. (2019). Detecting Pedestrians with YOLOv3 and Semantic Segmentation Infusion. In the 26th International Conference on Systems, Signals and Image Processing (IWSSIP 2019). Qualis CC: B1. Status: Submitted;

1.5 WORK ORGANIZATION

The structure of this work is served in the following organization: in Chapter 2 some concepts are briefly introduced such as Deep Learning, Convolutional Neural Networks, and Pedestrian Detection; a review of the related works is presented in Chapter 3, containing state-of-the-art techniques and being the main technical source for the studies executed in this work; the YOLOv3 is reviewed in a dedicated chapter (4); Chapter 5 explains the proposal; the experiments are described in detail in Chapter 6; results and general conclusions about the experiments are discussed in Chapter 7; the last chapter (8) presents an overview of the proposal, experiments and general results achieved in this work;

[3] goo.gl/anFCoc

2 THEORETICAL FOUNDATIONS

In this chapter, the fundamental concepts for the elaboration of this work are briefly presented. The tackled topics are Machine Learning, Neural Networks, Deep Learning, Transfer Learning, and Convolutional Neural Networks. Lastly, the tasks of Object and Pedestrian detections are reviewed with the explanation of its theory, goals, compositions, main approaches, and some challenges.

2.1 MACHINE LEARNING

According to Samuel (1959), Machine Learning is a Computer Science area which gives the computer the ability to learn without being explicitly programmed. To Mohri et al. (2012), the topic can be explained as utilizing the experience earned through data collection and then applied to computational methods to increase the predictions performance. This experience is related to the information available to the learning mechanism. The data can be digital and equivalent to a training dataset annotated by humans, where the quality and quantity of such annotations are directly related to the prediction performance.

Mitchell et al. (1997) gives a formal definition for the algorithms which fit in the Machine Learning concept: a computer program is said to learn from experience E with respect to some class of task T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

According to Mohri et al. (2012), variations in the available data types for training create some scenarios in Machine Learning according to the training and testing data used for evaluation. One of these scenarios is Supervised Learning, wherein the learning algorithm receives annotated training sample data, and make predictions for new data. These scenarios usually are associated with Classification and Regression tasks. The Non-Supervised Learning is another scenario, after which the learning algorithm only receives non-annotated data and make predictions over them. The Clustering task is related to this kind of situation. In the Semi-Supervised Learning, the algorithm gets both annotated and non-annotated training data, then make predictions over new data. This last scenario is related to the possibility of the non-annotated data distribution helping in the predictions. Nevertheless, other kinds of scenarios do exist: Transduced Inference, Online Learning, Reinforcement Learning, and Active Learning.

Machine Learning algorithms have been successfully applied to applications such as (Mohri et al., 2012): Text or document classification, e.g., *spam* detection; Natural language processing, e.g., morphological linguist analysis; Speech recognition and synthesis, and speaker verification; Optical Character Recognition (OCR); Computational biological applications, e.g., structured prediction; Computer Vision tasks, e.g., face detection; Fraud detection and network intrusion, e.g., credit card fraud detection; Games; Unassisted control of vehicles; Medical diagnoses; Recommendation systems, search engines, and information extraction systems;

Some learning task classes are defined to tackle applications like those cited above (Mohri et al., 2012):

- Classification: to classify a specific item in one of the known classes. For instance, classify the text category among the classes: sports, politics, weather, and others available to the learning algorithm;

- Regression: to predict a numeric value for a given item. For instance, predict parametric values for a stock exchange according to learned tendencies;
- Clustering: to divide the given items into homogeneous groups. For instance, identify implicit communities through social networks analysis;
- Dimensionality Reduction: to transform the representation of a specific item to a smaller dimensional version, preserving the primordial characteristics. For instance, Digital Image Processing.

2.1.1 Artificial Neural Networks

The Artificial Neural Networks (ANNs) have been initially proposed in the 1940s (Pitts and McCulloch, 1947) with the idea of simulating the human brain system to work on learning problems (Zhao et al., 2018). Mitchell et al. (1997) analogously describe ANNs as being constructed over interconnected sets of simple units, where each unit takes a number as input. Each input number comes from outputs of other units, producing a single output value. Such description matches the human brain neural network composition, where each unit imitates a neuron. The artificial neuron in an ANN will take many inputs and generate a single output, while each neuron is interconnected with others in a layer scheme (Sharma et al., 2012). Each neuron produces activations based on the environment or weighted connections from other neurons, creating a chain of activations that brings the input data from the initial layer to the desired output value in the final layer (Schmidhuber, 2015).

The popularity of ANNs has increased in the 80s and 90s by the introduction of the backpropagation algorithm. Due to the overfitting problem, lack of training data, limited computational capacities and low general performance, the research community had decreased the interest in the area. They started gain back attention with the rise of the Deep Learning. The later was enabled by the emergence of large-scale datasets, such as ImageNet (Krizhevsky et al., 2012); the progress in high performance computing powered by Graphics Processing Units (GPUs); and significant development of network designs (Zhao et al., 2018).

The "Perceptron" is one the most basic structures introduced in ANNs back in the 1950s. It can be considered as a device which makes decisions by "pondering up" pieces of evidence (Mitchell et al., 1997). It takes binary inputs and produces binary outputs (Figure 2.1). For each input, there is a weight that regulates the relevance of the incoming data to the output. The "Sigmoid Neuron" is a different structure but similar to the Perceptron. Instead of accepting just binary values, it is taken any value between 0 and 1. The Sigmoid Neuron also has the weights for each input and a "bias" value as well (Nielsen, 2015).

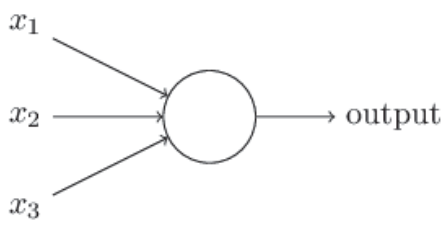


Figure 2.1: Perceptron. Source: (Nielsen, 2015).

The neural networks are formed by three main layers (groups) of neurons (units): input, hidden and output. Figure 2.2 illustrates the common network composition. The input layer is where the data is initially inserted. The hidden layer has its neurons connected to the ones in the

input layer, being free to define their representation of the input data. In the output layer, the neurons will be finally activated according to the neural path used in the hidden layer (Baba et al., 2013).

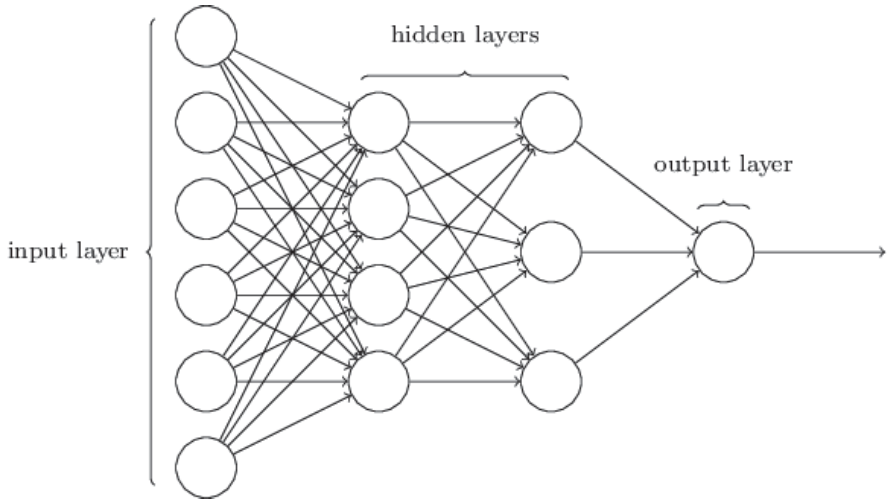


Figure 2.2: Neural Network composed of multiple layers, also known as "multilayer perceptrons." Source: (Nielsen, 2015).

There are two main types of neural networks. The "feed-forward network," also known as "non-recurrent network" which allow the activations to occur in a single direction (one-way only), that is, from the input layer to the output. This kind of network is simple and commonly used in the Pattern Recognition area. The second type is the "recurrent network" or "feedback network." In this type, the activations are propagated in both directions creating loops inside the network (Sharma et al., 2012).

The training process of a neural network, for instance to the classification task, aims to teach the model to receive inputs and correctly assign a wanted output. So it is necessary to have an algorithm that helps in defining the weights and bias of all neurons in the network in such way that the during the learning process the network's output starts approximating to the desired prediction according to the inputs. The "cost functions" (a.k.a., "loss functions") have been created to calculate the learning progress in function of the inputs, weights, bias and desired output. The "quadratic cost function" (a.k.a, "mean squared error") is a popular loss function (Equation 2.1). The lower the value calculated by the cost function is, closer the model is in predicting the aimed output. The algorithm which takes the parameters (weights and biases) and minimizes the cost value is called "Gradient Descent" (Nielsen, 2015).

$$C(w, b) \equiv \frac{1}{2n} \sum_{x} \left\| y(x) - a^L(x) \right\|^2 \tag{2.1}$$

where C is the cost function; w is the weight value; b is the bias value; n is the number of training examples; $y = y(x)$ is the desired output; x are the individual training examples; L is the number of layers in the network; and $a^L = a^L(x)$ is the vector of activations of the network's output for an input of x.

The "backpropagation" algorithm has been introduced to compute the gradient of a cost function. This algorithm still is an essential component in the neural networks learning process, even that it was initially proposed back in 1970. The primary mission of the backpropagation algorithm is to calculate the partial derivatives present in the cost function according to the weights and biases from the neural network. In other words, it needs to figure out the variation

in the error according to the changes in the parameters, for instance, increasing or decreasing the weights values. It is mostly based in common linear algebraic operations, i.e., matrix multiplications (Nielsen, 2015).

The multilayer networks contain an "error surface" with possibly multiple "local minimas". That is, regions in the parameters space which the error is lower, but not necessarily the lowest. The backpropagation algorithm, using the gradient descent, searches in the parameters space for the "global minimum error." It corresponds to the lowest error the gradient descent algorithm could ever encounter among all local minimas. However, the gradient descent does not guarantee to find the global minimum error (best local minima), once it can get trapped in some local minima according to the algorithm mechanics regarding the dimensional space provided by the parameters. As more parameters are in the network, more escape routes will be available in the error space for the gradient descent to continue minimizing the error (Mitchell et al., 1997). Figure 2.3 illustrates the "path" chosen by the Gradient Descent algorithm, trying to avoid getting stuck in local minimas and going in the direction to the minimum global error.

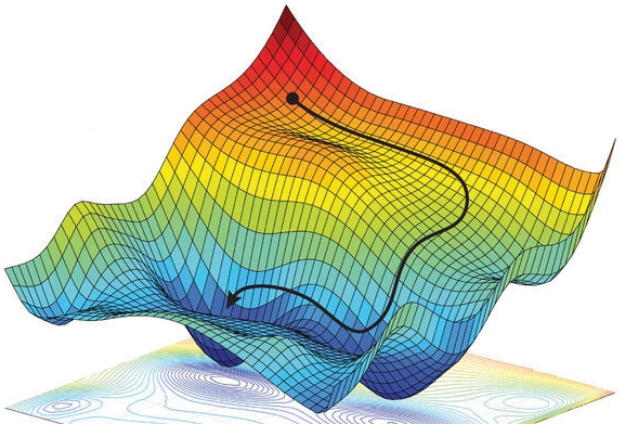


Figure 2.3: Gradient Descent convergence to a minimum error, in an error space. Source: Alexander Amini, Daniela Rus, Massachusetts Institute of Technology. Adapted by M. Atarod/Science (www.sciencemag.org).

Mitchell et al. (1997) summarize the general learning idea: the backpropagation algorithm finds the parameters of a multilayer network, by applying the gradient descent algorithm for minimizing the error calculated by the loss function according to the network predicted output against the desired value.

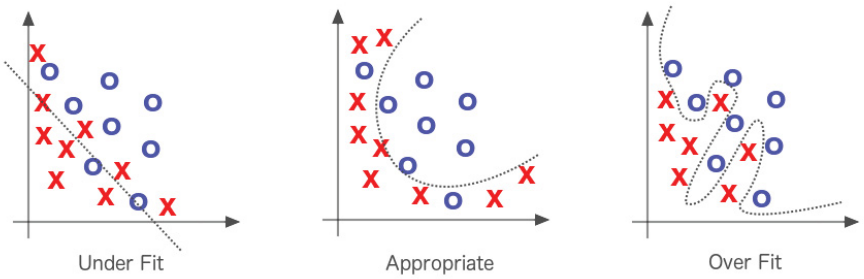


Figure 2.4: Example of classification task presenting cases of underfitting (left), ideal model complexity (center), and over-complex model (right). The lower the model complexity, harder is to the model to correctly separate both classes (left). However, if the model gets too complex, it becomes too specialized in the training data which reduces the model's generalization power (right). That is, the model will struggle to correctly classify new data, even from the same context. The ideal model (center) would need to have similar performance for the training and the testing data, being not too specialized and not too loose to the data. (Patterson and Gibson, 2017)

While training a neural network, the "generalization problem" is one of the most common challenges. A network with good generalization will achieve good results both in the trained situation (training data) and a new situation (new data). The concept is to gather just the right amount of knowledge from the data, neither too much (overfitting) nor too little (underfitting) (Figure 2.4). The primary strategy in accomplishing good generalization levels is by finding the simplest model possible that well represent data. Reducing the complexity of the model will also reduce the number of errors. There are a few methods to help to achieving good generalization. One is the "Early Stopping" which aims to stop the training before the Gradient Descent finds the global minimum error. It is accomplished by not using all weights. Consequently, that decreases the complexity of the model. If the number of the iterations are too high, more of the weights will be tuned until the model fits the training data in the slightest details, and then it may not understand new data. Another technique is the "Regularization." This method seeks to penalize the model complexity, by controlling the weights values to produce "smoother" functions. It is similar to reducing the number of neurons in the network, once minimizes the size of the slopes in the function which tends to overfit the data (Hagan et al., 1996).

2.2 DEEP LEARNING

The Deep Learning is a subarea of Machine Learning, formulated with algorithms able to learn in multiple representation levels, which establish a higher complexity relationship between the data (Deng et al., 2014). Its main characteristic is that the feature extractors automatically learn through the data from generic learning procedures (LeCun et al., 2015).

According to LeCun et al. (2015), conventional Machine Learning techniques are limited regarding its abilities to process data in more natural formats. In this case, the data needs to be processed and fed to the algorithms in such way that exactly correspond to the calculation requirements, demanding a meticulous project for feature extraction that would be able to transform original raw data into a set of meaningful features.

The Deep Learning allows the data to be inserted in the projected network in a more natural format, which will automatically detect the relevant data representation for tasks like Classification and Detection, for example. Calculation procedures are accumulated in many layers, wherein each subsequent layer makes the data representation more abstract. This technique enables the models to learn complex functions that amplifies the relevance of essential features and suppress the less relevant ones (LeCun et al., 2015). With the advent of Deep Learning, the development of new models can have the efforts transferred, now, to the design of better network architectures instead of spending efforts in searching for a good feature representation (Liu et al., 2018).

According to Liu et al. (2018), in the last years, Deep Learning has promoted significant progress in a broad variety of applications such as Visual Recognition, Object Detection, Speech Recognition, Natural Language Processing, Medical Image Analysis, Drug Discovery, and Genomics. The increasing development of the area has been mainly supported by powerful computing resources, such as GPUs and the availability of large-scale datasets which are essential when training Deep Learning models. The crescent growth in processing power brings the feasibility to the training time, while the datasets are enabled to feed more complex training data with higher variety and quantities among the samples.

2.3 CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks (CNNs), also known as *ConvNets*, are a kind of network related to the majority of works that reached the state-of-the-art in the Computer Vision area (Szegedy et al., 2016; Huang et al., 2017; Redmon and Farhadi, 2017; Ren et al., 2015). For Zhao et al. (2018), CNN is the most representative model in Deep Learning. According to Szegedy et al. (2015), CNNs are also used in many other works applied to specific tasks like Object Detection, Segmentation, Pose Estimators, Video Classification, Object Tracking, and Super-resolution.

Goodfellow et al. (2016) explain that a CNN is a neural network which one or more layers apply the mathematical operation called "convolution" instead of the default matrix multiplication.

A CNN is composed by the combination of specific neural network layers, typically formed by three main stages: convolution, non-linear activation function, and pooling function. The convolutional layer applies filters (kernels) convolutionally in windows for an image, generating a new image (feature map) which by default has its dimensions slightly reduced, where the detected features are accentuated (Figure 2.5). In the second stage, each linear activation generated in the convolution is submitted to nonlinear activation functions, such as the Rectified Linear Unit (ReLU). By introducing the non-linearity into the model, it enables better gradient propagation. The last stage statistically summarizes the most relevant features according to neighboring outputs from the previous layer. For instance, the "max polling" layer takes the maximum value determined by windows from the previous output (Goodfellow et al., 2016).

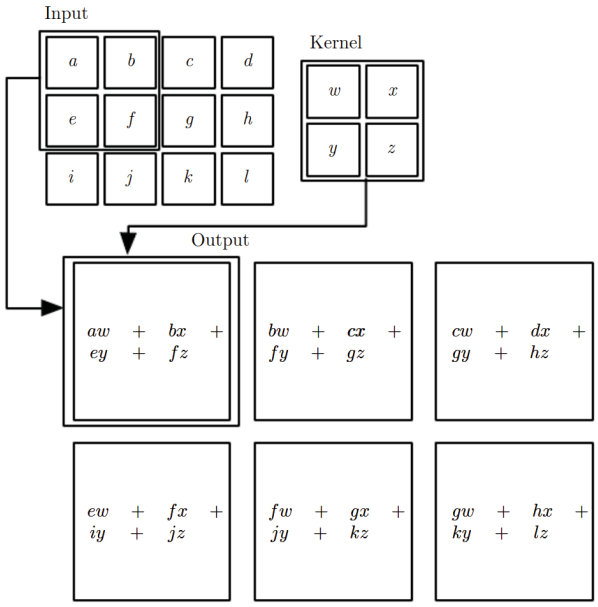


Figure 2.5: Example of a convolutional operation. The input is a 2-D matrix 3×4 and there is a kernel as a matrix 2×2. In a first step, a window of the same size of the kernel is selected in the input. The elements in both window and kernel are multiplied and summed, generating a single output value. The window is dislocated a single unit per step until covering the entire image, repeating the convolutional operation in each step. At the end of the operation, there will be a resulting matrix of 3×2 which represents a feature map. Source: (Goodfellow et al., 2016).

Each layer in a CNN can be referred to as a "feature map." Usually, the input in the network is an image represented by a three-dimensional (3D) matrix where each dimension is one color channel in the case of RGB format. For black and white images there is a single dimension. Each pixel from the image is represented by a position in the initial layer which is equivalent to a specific feature (Zhao et al., 2018).

According to LeCun et al. (2015), this kind of network provides a high level of invariance to scale, distortion and displacement. Figure 2.6 demonstrates a simple CNN model utilized by LeCun et al. (1998) in a digits recognition work.

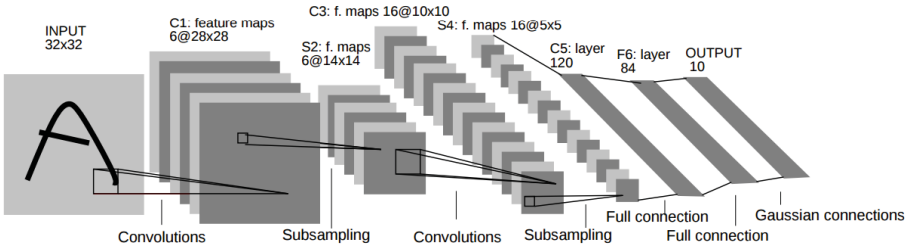


Figure 2.6: LeNet-5 CNN architecture utilized in the task of digits recognition. Source: (LeCun et al., 1998).

The CNN can learn alone the weights for the filters used in the feature extraction. The neurons involved extract visual features such as edges, end-points, corner, and others. The combination of those features creates new ones of a higher level that are used in the subsequent layers (LeCun et al., 1998).

According to Simonyan and Zisserman (2014) and related works such as (Krizhevsky et al., 2012), CNNs with many layers end up creating the category of DCNNs (also known as Deep ConvNets). Simonyan and Zisserman (2014) demonstrate that increasing the number of convolutional layers has been providing better model performances in many works. Figure 2.7 is an example of a DCNN named "Inception v3" (Szegedy et al., 2016), wherein it is possible to see its multiple convolutional layers in yellow. According to Liu et al. (2018), DCNNs has driven outstanding results in processing images, videos, speech, and audio.

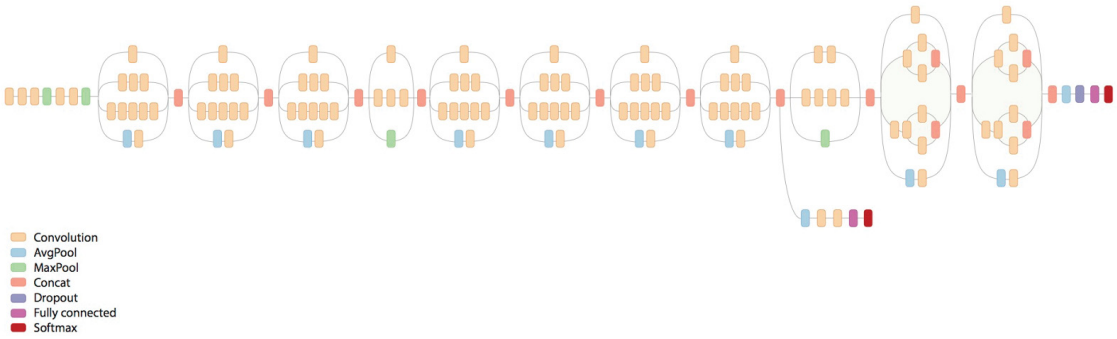


Figure 2.7: Network structure of the model Inception v3. Source: Google AI Blog - Train your own image classifier with Inception in TensorFlow[1].

2.4 TRANSFER LEARNING

Transfer Learning, according to Pan and Yang (2010), has as the main idea the utilization of knowledge already obtained in specific domains and then applying this knowledge to solve problems in different domains. This learning process tends to be faster than building knowledge from scratch.

[1]https://research.googleblog.com/2016/03/train-your-own-image-classifier-with.html

Pan and Yang (2010) explain that for Machine Learning algorithms the knowledge extracted from training data in a given domain is not restricted to be applied only in the same specific space of features. Also, it does not need to be in the same distribution in order to allow the reuse of the knowledge for other applications. In real scenarios, it is common to have an insufficient amount of data for the training process when tackling some problem. At the same time, usually there is abundant data for other domains, differing only in data distribution and the features set (Junlin et al., 2016).

In the Machine Learning area, the Transfer Learning can be defined as the process of applying fine-tuning and reusing a pre-trained neural network in a different domain from the one it has been initially prepared. One of the benefits of using Transfer Learning is that the fine-tuning process is relatively quicker than training a model from scratch, and the reused weights are generally of high quality when trained in big datasets composed by high variances within the classes, such as the ImageNet[2] dataset which contains one thousand classes and 1.2 million images.

The practical proof of the benefits of Transfer Learning can be easily obtained with experiments publicly available, e.g., the "TensorFlow for Poets"[3], wherein the model "Inception v3" (Szegedy et al., 2016), pre-trained over ImageNet, receives a fine-tuning for a task of flower classification, being in a different domain from what it has been initially trained.

2.5 OBJECT AND PEDESTRIAN DETECTION

According to Zhao et al. (2018), in the last years, object detection has been attracting attention for researching due to its significance in video analysis and image understanding. Object Detection corresponds to the task of precisely estimating the location and the categories of objects in a given image. For Liu et al. (2018), Object Detection form the basis for many Computer Vision tasks of higher complexity, such as object tracking, event detection, and scene understanding. Among the possible applications are robot vision, autonomous driving, and surveillance.

When an object detection algorithm receives an image as input, it must find and present as a result the spatial locations of every instance of any known object class in the figure or video frame. Locating the object in the image is known as “object location” and determining its category (class/type) is called “object classification.” However, there are many challenges related to this task. An object detector with high-quality detections must be able to recognize objects in a large variety of real-world categories, having “high distinctiveness.” Another objective is in having “high robustness,” that is, being able to understand that there are categories of objects which have high intra-class appearance variation and to predict correctly the same respective category to them. In general, object instances present high variance to appearance such as illumination, a wide variety of poses, scale, occlusion, motion, blur, and others (Figure 2.8). The last challenge is related to the detection efficiency. Due to the exponential increase of images in social media and mobile devices, detectors are required to scalable and efficient (Liu et al., 2018).

According to Zhao et al. (2018), the traditional pipeline for object detection can be defined in three main steps: 1) the selection of informative regions: the whole image is scanned and them all possible positions for objects are established; 2) extraction of visual features: it is necessary to have semantic and robust representation of the objects in order to recognize them, and that is done by extracting features from the informative regions; 3) the classification: in order to understand in which object class an object belongs, a classifier is responsible for distinguishing the representative features obtained in the previous steps.

[2]ImageNet: http://www.image-net.org/

[3]TensorFlow for Poets: https://codelabs.developers.google.com/codelabs/tensorflow-for-poets

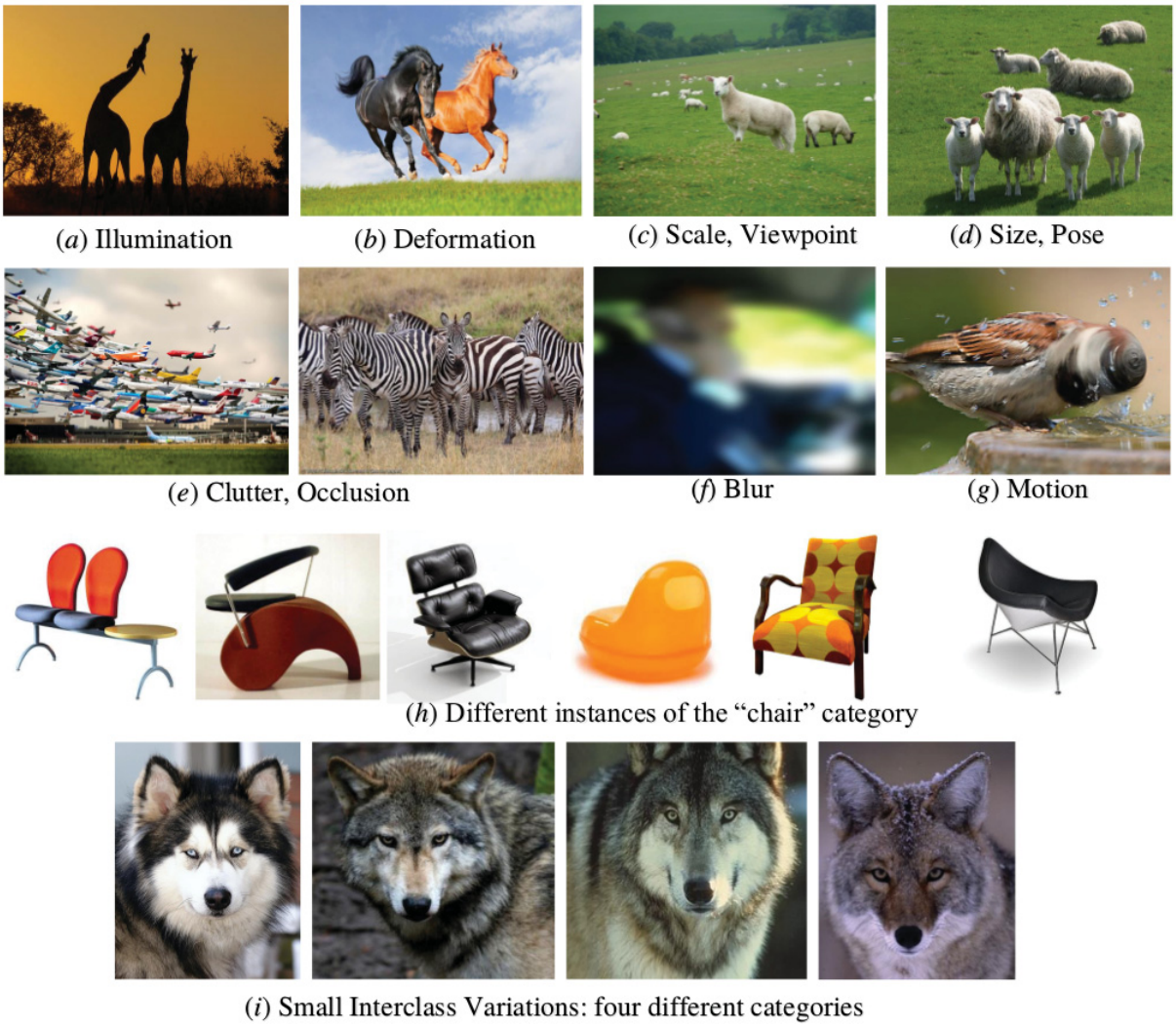


Figure 2.8: Summary of challenges in appearance for the Object Detection. Source: (Zhao et al., 2018)

Initially, such detection methods were powered by handcrafted feature extractions and shallow network architectures which rapidly stagnates regarding performance (Zhao et al., 2018). One of the very first works with Pedestrian Detection was made by Viola et al. (2003) using Haar-like features and a cascade of classifiers. Lately, Deep Learning methods have enabled the automatic feature representations learning directly from the data, providing great progress in the object detection area (Liu et al., 2018). Zhao et al. (2018) present in a timeline (Figure 2.9) the milestones containing main works in Object Detection, feature representations, frameworks, dataset, and other information. It is possible to see that since 2012, when Krizhevsky et al. (2012) introduced the AlexNet (DCNN technique) and promoted outstanding results, efforts in the computer vision started to become more directed to Deep Learning methods.

There are many works in Object & Pedestrian Detection which make use of DCNN. This kind of network has been promoting progress in this detection area. The Faster R-CNN (Ren et al., 2015) is an example of a DCNN based method. It works in two stages, wherein in the first one a "region proposal network" (RPN) generates a large number of regions in the image that are found likely of having an object. In the next stage, each of the proposed regions is classified in order to define the object's class. Such approach has made Faster R-CNN as a baseline for many works in this same area or to related problems, being known for its wide variety of derivative works and success in detection tasks (Brazil et al., 2017).

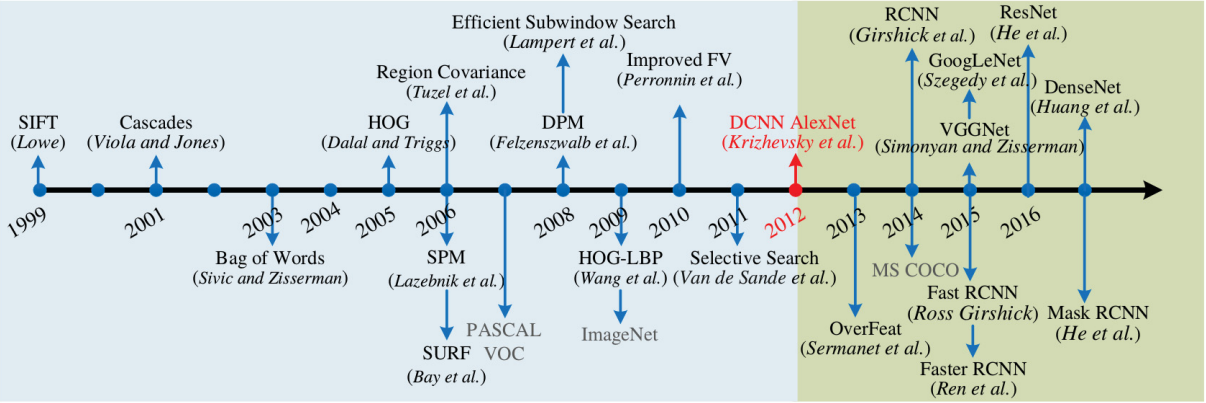


Figure 2.9: Milestones in Object Detection since 1999. There is works related to feature representations, frameworks, dataset and other. From 2012, since AlexNet, works started to focus on Deep Learning techniques. Figure adapted from Zhao et al. (2018).

Pedestrian Detection is a sub-task of Object Detection (Zhao et al., 2018). Figure 1.1 demonstrates examples of object and pedestrian detections. It is one of the most explored problems of object detection due to its real-world importance (Brazil et al., 2017), being applicable to surveillance, transit safety, and robotics, for example (Correia, 2016). The main challenges are related to detection speed (Varga and Szirányi, 2017) and detection quality which is mainly affected by substantial changes in the pedestrian appearance (Liu et al., 2018). Pedestrian Detection is closely related to other topics like Pedestrian Tracking, Person Re-Identification and Robot Navigation (Zhao et al., 2018).

Pedestrian detection can also be referenced as Person Detection, but they are slightly different. While Person Detection is about detecting people in general independently of pose, appearance or context, Pedestrian Detection focuses on people standing, walking or running. The latter is more related to people on flow in the scenario, like people on transit, crowed or video-surveillance for instance (Dollár et al., 2012).

According to Brazil et al. (2017), there are two main Computer Vision approaches for finding pedestrian in an image. The first is through object detection which has as strength the excelent performance in distinguishing distinct objects, but at the same time does not provide much information about the object's shape. The second approach is using semantic segmentation, wherein does give useful information about the object's boundaries but has as a weakness the difficulty in separating instances of the same class. Both techniques establish such natural contrast between each other. That is why most of the pedestrian detection task inherits from the object detection methods. However, the segmentation technique has been utilized as additional and complementary method such in works like from Du et al. (2018, 2017) and Brazil et al. (2017).

Liu et al. (2018) define two main categories for object detection frameworks. The first category is called "region-based" or "Two Stage Framework." This category refers to the process of selecting regions in the image which are, at first, class-independent. That is, these regions do not contain the definition of which category the probable object belongs to, it merely looks for objects of any kind. In the next step, features are extracted through the use of CNNs. In the last step, a classifier will label each region to distinguish different object classes. Such approach has been initially proposed in the RCNN work by Girshick et al. (2014), concurrently with other methods. It inspired many successors over the years such as the wide-explored Faster RCNN (Ren et al., 2015) and the recent Mask RCNN (He et al., 2017).

The second category defined by Liu et al. (2018), is a single stage framework which corresponds to a "Unified Pipeline" (a.k.a. "One Stage Pipeline" or "region proposal free"). This

kind of framework does not contain a separated component for the candidate regions generation. The entire detection process, since the input of the image, feature extraction and prediction is part of a single pipeline. Among the current methods which implement their algorithm with this kind of framework structure, YOLO (Redmon et al., 2016) is one of the most notorious ones, which currently has reached its third version (Redmon and Farhadi, 2018). The authors implemented the single-pipeline making YOLO look the entire image at once to generate the features, detect and classify, all in the same network. That is, contrary to the region-based, YOLO takes in account the context of the entire image when calculating the features, while in the region-based methods proposals of object locations are generated by evaluating in an isolated manner individual parts of the image. Another difference in the single-stage framework compared to the two-stage one is that the former do predict the "objectness" score of a probable location of an object and at the same time it does predict its class. In the two-stage framework, those are independent steps with isolated knowledge.

3 BIBLIOGRAPHIC REVIEW

After the introduction of the Object and Pedestrian Detection concepts in the previous Chapter, in this section, we briefly review a few popular pedestrian detection methods applied in the early days and five works that currently hold the top-tier techniques in the Caltech Pedestrian Detection Benchmark.

3.1 PEDESTRIAN DETECTION

According to Benenson et al. (2014), the general methods used in Object Detection have been based on four main concepts: "Viola & Jones" derivatives, Histograms of Oriented Gradient (HOG)+Support Vector Machine (SVM) templates, Deformable Part Model (DPM) and CNNs.

Varga and Szirányi (2017) explain that Papageorgiou and Poggio (2000) have introduced one of the first works to tackle Pedestrian Detection, which they proposed the use of a dictionary of Haar wavelet features together with SVM. Viola et al. (2003), introduced the used of AdaBoost to train a novel detector based on the framework from Viola and Jones (2001), considering at the same time motion and appearance information. It had achieved the capability of fast detection (for the time), detecting small pedestrians and with a low rate of false positives. Dalal and Triggs (2005) proposed the Histograms of Oriented Gradient (HOG) becoming a very explored method and basis of many works, and in the same work, they did introduce the still utilized and well-known pedestrian dataset called "INRIA." Dollár et al. (2009a) introduced the Integral Channel Features (ICF) within the use of various types of features, which became a successful technique applied to many pedestrian detectors. Felzenszwalb et al. (2010) proposed a detector based on HOG and named "deformable part based model" (DPM) which achieved outstanding results in pedestrian detection at the time.

Before the popularization of Deep Learning techniques, the most successful pedestrian detectors were mainly based in combined boosted decision forest with hand-crafted features, and also part-based models that focused in solving the deformation and occlusion problems, such as the DPM (Zhao et al., 2018). From 2012 on, influenced by the outstanding results obtained in the image classification challenge ILSRVC by the proposed DCNN called "AlexNet" (Krizhevsky et al., 2012), many works started to invest in such Deep Learning methods (Liu et al., 2018). However for Pedestrian Detection, according to Zhao et al. (2018), during a long time DCNNs could not overcome the best hand-crafted feature technique (the "Checkerboards+" method by Zhang et al. (2015)) even by applying the part-based and occlusion techniques (Tang et al., 2014).

Zhao et al. (2018) review a few of the works that have been proposed in such a period when the researchers had been attracted to challenge the shallower methods. Zhang et al. (2016a) proposed an adaptation of a generic object detection method called "Faster R-CNN" (Ren et al., 2015), by focusing the Region Proposal Network (RPN) in tackling the hard negative samples and small instances, and by incrementing the features from the high resolution convolutions with the boosted forests. The DeepParts (Tian et al., 2015), proposed as a Deep Learning framework, has been introduced by an ensemble of extensive part detectors, which was based in the well-known DPM method (Felzenszwalb et al., 2010). DeepParts achieved its results by the sophisticated combination of 45 fine-tuned DCNNs models and other techniques. Hu et al. (2018) prepared an ensemble of boosted decision models with simple pixel labeling, complementary hand-crafted features and reusing convolutional feature maps. CompACT-Deep (Cai et al., 2015) proposed

a complex combination of hand-crafted features with fine-tuned DCNNs. Going towards less complex architectures, Tomè et al. (2016) proposed the optimization and adaptation of generic object detection, and later Tomé et al. (2016) also proposed a reduced memory region based DCNN using Aggregated Channel Features (ACF) detectors and SVM classifiers.

In addition to their review, Zhao et al. (2018) also did evaluate some works using the Caltech Pedestrian Dataset. This dataset is very popular among the pedestrian detection task, containing images from a moving vehicle in an urban environment. They test 7 methods based on DCNN and the other two implemented with hand-crafted features. In this evaluation, they verified that in general, the DCNNs works did perform better than the hand-crafted ones. The DeepParts (Tian et al., 2015), despite good results, has high processing costs and consequently, it is very time-consuming. Among the experimented methods, the best performance is achieved by F-DNN-SS (Du et al., 2017). Zhao et al. (2018), concludes that Deep Learning methods achieve the best accuracy performances, and the hand-crafted features are now relevant when used as some complementary clue to be combined with other CNN features.

Dollár et al. (2012) have reviewed some pedestrian/person datasets. According to them, there are two kinds of datasets: one for "person" and the other for "pedestrian" detection. The first is related to the detection of people in any context or situation. There is no specific domain, pose or other restriction, corresponding to daily scenes. LabelMe (Russell et al., 2008) and PASCAL VOC (Everingham et al., 2010) are examples of this kind of dataset. However, the pedestrian datasets are related mostly to the upstanding pose, and commonly, people in movement in the scene. The latter kind is composed by a bigger number of datasets (Dollár et al., 2012; Benenson et al., 2014), such as INRIA (Dalal and Triggs, 2005), ETH (Ess et al., 2007), TUD-Brussels (Wojek et al., 2009), Daimler (Wojek et al., 2009), Caltech (Dollár et al., 2009b), KITTI (Geiger et al., 2012) and others[1].

Among the many available datasets, Table 3.1 summarizes some relevant information about a few of them. According to Benenson et al. (2014), INRIA is another pedestrian dataset that is commonly chosen for training due to its high-quality annotations and because of its variety of scenes in which the person appears. It is also among the oldest ones, which contains much fewer images compared to most recent ones, and do not correspond to video frames. Benenson et al. (2014) also state that together with Caltech, KITTI (Geiger et al., 2012) compose the most commonly experimented datasets in pedestrian detection. KITTI is known by the diversity in its test set, and by its evaluation toolbox that remains with a set of unpublished testing data. The Caltech is a widely explored dataset, composed of images of a moving vehicle on urban scenarios (further description at Chapter 6). CityPersons Zhang et al. (2017) is a new dataset for pedestrian detection, having been based on the CityScapes (Cordts et al., 2016). Compared to the Caltech dataset it corresponds to more crowed scenes with a higher number of occurrences of occlusion.

Table 3.1: Comparison between a few pedestrian datasets. Some information was obtained from (Dollár et al., 2012) and other directly from each dataset works.

Dataset	Year	Images	Pedestrians	Resolution	Aquisition	Type	Authors
INRIA	2005	2,573	1,774	<96x160	photo	Unconstrained environments	Dalal and Triggs (2005)
KITTI	2012	14,999	~22,000	1392 x 512	static	Moving vehicle on urban cenario	Geiger et al. (2012)
MOT17Det	2016	11,235	295,163	1920x1080	static & mobile	Busy urban pedestrian zones	Milan et al. (2016)
CALTECH	2009	249,000	347,000	640x480	mobile	Moving vehicle on urban cenario	Dollár et al. (2009b)
ETH	2007	2,303	14,388	640x480	mobile	Busy urban pedestrian zones	Ess et al. (2007)
TUD-Brussels	2009	1,818	3,274	520x756	mobile	Busy urban pedestrian zones	Wojek et al. (2009)
Daimler-DB	2009	28,500	72,100	640x480	mobile	Moving vehicle on urban cenario	Enzweiler and Gavrila (2009)
CityPerson	2017	5000	35016	~2040x1016	mobile	Moving vehicle on urban cenario	Zhang et al. (2017)

[1]http://www.cvpapers.com/datasets.html

3.2 RECENT WORKS

In this section, we briefly review the top-5 pedestrians detectors reported by the Caltech Pedestrian Detection Benchmark using the "Reasonable" metric. More information about the dataset and its default metric is available in Chapter 6.

3.2.1 F-DNN+SS

Du et al. (2017) proposed the first version of a fused architecture for pedestrian detection called Fused Deep Neural Network (F-DNN) (Figure 3.1). A pedestrian proposal generator composes it through a DCNN which produces a large number of candidates and consequently high quantities of False Positives. This generator is a single-shot multi-box detector (SSD) (Liu et al., 2016), configured to use a low confidence score threshold.

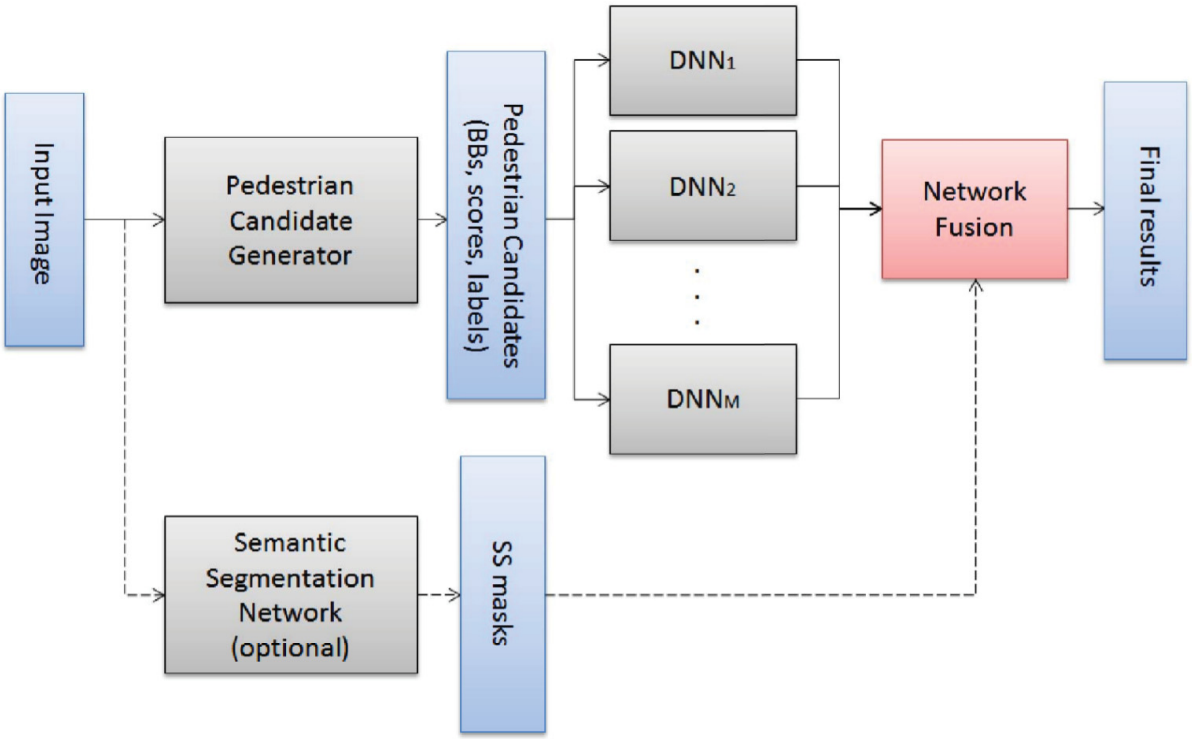


Figure 3.1: F-DNN+SS composed by a pedestrian candidate generator, network fusion method that handles multiple neural network classification, and a parallel semantic segmentation network. Source: (Du et al., 2017)

A separated and parallel network composed of multiple Deep Neural Network classifiers is connected to a fusion method. This method filters the pedestrian candidates through a soft-rejection based network and is called "soft-rejection based network fusion" (SNF). The task executed in this step aims to make the classification decision by aggregating the confidences from the pedestrian candidates incoming from each classifier. This is proposed instead of the hard binary classification, which accepts or rejects a candidate.

Additionally, the authors exploit segmentation as a strong clue in the framework by the introduction of an additional parallel network based on an adapted version of the VGG16 network (Simonyan and Zisserman, 2014). They call it "context aggregation dilated convolutional network with semantic segmentation" (SS). This network predicts segmentation masks which are fused with the other networks through the ensemble, using deep dilated convolutions and context aggregation. The masks are binary, and the activated pixels correspond to pedestrians locations. They are used for post-processing, suppressing the bounding boxes which do not

contain pedestrians, by providing an additional vote for the soft fusion metric. Due to the lack of semantic segmentation annotation for the most pedestrian datasets, the authors used the recent CityScape (Cordts et al., 2016) dataset to train the model.

The authors evaluate at least two different network compositions over the Caltech dataset. The first (F-DNN) does not contain the additional segmentation network. The second (F-DNN+SS) include the semantic segmentation task which despite achieving high-quality accuracy it degrades the detection speed from 0.30 to 2.48 seconds per image, and requiring multiple GPUs. At the time, the method outperformed the competitors both in accuracy and speed. The authors did not reference the source code availability.

3.2.2 F-DNN2+SS

The authors from F-DNN+SS (Du et al., 2017) released the second version of their framework, now called F-DNN2+SS (Du et al., 2018). This version contains a few improvements compared to the former one, increasing the accuracy performance over the evaluated datasets. Instead of using only the Caltech dataset (Dollár et al., 2009b), the authors now evaluate the detector using three additional ones: INRIA (Dalal and Triggs, 2005), ETH (Ess et al., 2007) and KITTI (Geiger et al., 2012). The authors have also prepared the model to handle more classes, such as cars and cyclists.

The first improvement applied to this updated version refers to the application of a soft-label method for training the classifiers and learning the soft-rejection parameters by an additional fusion network. Compared to the first version, they propose a more sophisticated logic in the voting process. A floating point label is assigned to each pedestrian candidate according to the one with the best overlapping score compared to the ground-truth. Such logic is used in mild-confidence cases, that is, when the candidate confidence is too close to the threshold. Other than that, the hard-label (binary vote) method is used to define the candidate as pedestrian or background directly.

The authors also demonstrate that it is possible to train the classification system using two different methods. The first training method is the "end-to-end," which enables to train all the classification networks at once. To accomplish that, the loss layers are removed, and the outputs of each network are concatenated as an input layer for the fusion network. However, this configuration tends to lead to overfitting, and because the networks are different, they would require different hyper-parameters, and with different converging speeds. The second training method consists in firstly training the classification networks and then training the fusion network. The latter method is the chosen one by the authors, according to them due to being easily implementable.

Another improvement applied by the authors corresponds to a new method for fusing the semantic segmentation results with a pedestrian-shaped kernel. For every bounding box (pedestrian proposals) predicted by the candidate generator, a pixel-wise analysis is executed. A kernel is created through the mean of all ground-truth semantic segmentation binary masks and used as a weight factor for the soft fusion. The kernel will boost the scores of bounding boxes that match the general shape of the class, and otherwise, it will be decreased. Such a method is illustrated in Figure 3.2.

3.2.3 GDFL

Lin et al. (2018) submitted the most recent method in the Caltech Pedestrian Detection Benchmark. They propose to use graininess-aware deep feature learning (GDFL) that provide an attention

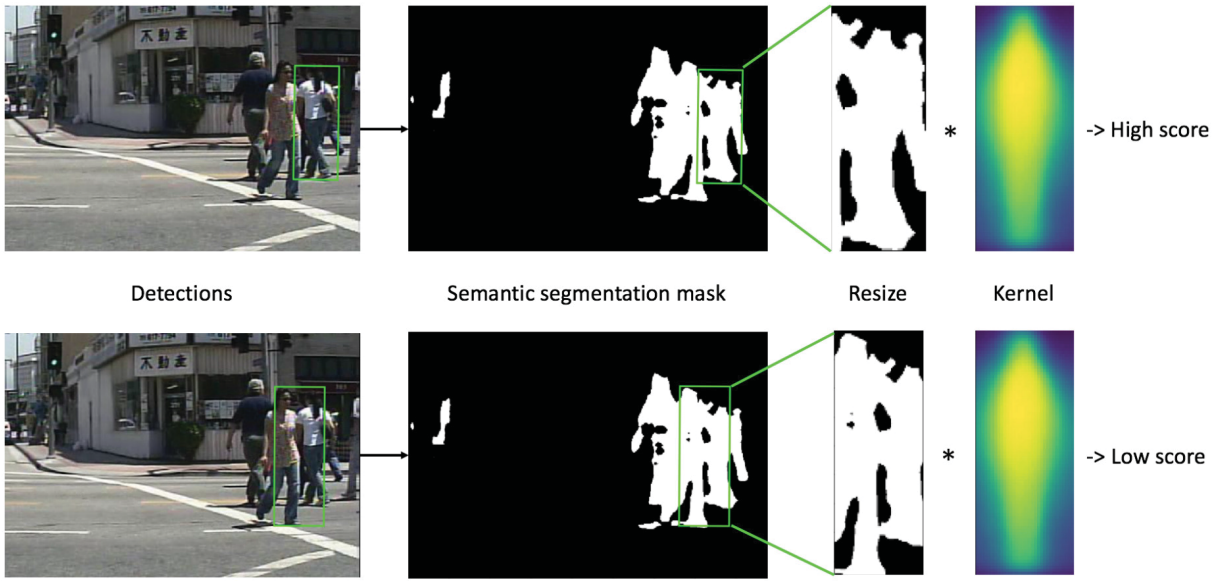


Figure 3.2: F-DNN2+SS kernel-based method. The detection systems are composed by the fusion of semantic segmentation clues, where a pedestrian-shaped kernel weights each prediction. Source: (Du et al., 2018)

mechanism which directly encodes feature maps highlighting pedestrians. They achieved outstanding accuracy and detection speed in the evaluated benchmarks.

Contrary to SDS-RCNN (Brazil et al., 2017) they do not use implicit supervision for the segmentation masks, and also they do not execute post-processing procedures with the masks as F-DNN+SS (Du et al., 2017) does. Instead of considering only low-resolution feature maps, they feed fine-grained convolutional features attached to human appearance. Such mechanism puts attention in the network for the pedestrian locations, decreasing the importance of regions in the image with no pedestrians (background). The attention masks are pixel-wise, and they determinate the probability of each pixel belonging to a pedestrian. The resulting features become very useful in detecting small pedestrians and occluded ones as well. This characteristic is enabled by the fine-grained feature enhancement which provides great ability in distinguishing pedestrians (foreground) from the background (remaining of the image).

Additionally, they propose a "zoom-in-zoom-out" method which works by inserting in the features information about local detail and context. This last method helps to improve even more the capability of detecting small pedestrians. It has been inspired by the natural process of human annotators having to zoom in and out on an image in order to better understand the detail of the human shape (zoom in) as well perceiving the context that involves the pedestrian in the scene. The projected detector is single-staged and trainable in an "end-to-end" way, that is, a monolithic pipeline with no extra procedures out of the main network. Additionally, GDFL demonstrated to be faster than the competitive methods.

Lin et al. (2018) present their detection framework formed by three main components (Figure 3.3). The first is a CNN backbone that generates multiple feature maps for pedestrians in different scales. The scale-aware attention module is the second component, which is responsible for generating attention masks which are injected into the previous feature maps, resulting in the referenced graininess-aware features. The last refers to the "zoom-in-zoom-out" module that will sum up more information to the features of the pedestrians in such different visualizations.

The authors evaluate their introduced method over four datasets: Caltech (Dollár et al., 2009b), INRIA (Dalal and Triggs, 2005), KITTI (Geiger et al., 2012) and MOT17Det (Milan et al., 2016). Among the evaluations, they report achieving 0.05 seconds (50 milliseconds (ms)

or 20 Frames per Second (FPS)) for running a detection over a single image, which is faster than the compared methods. However, the general network structure is not precisely described, and there are no references of source-code availability.

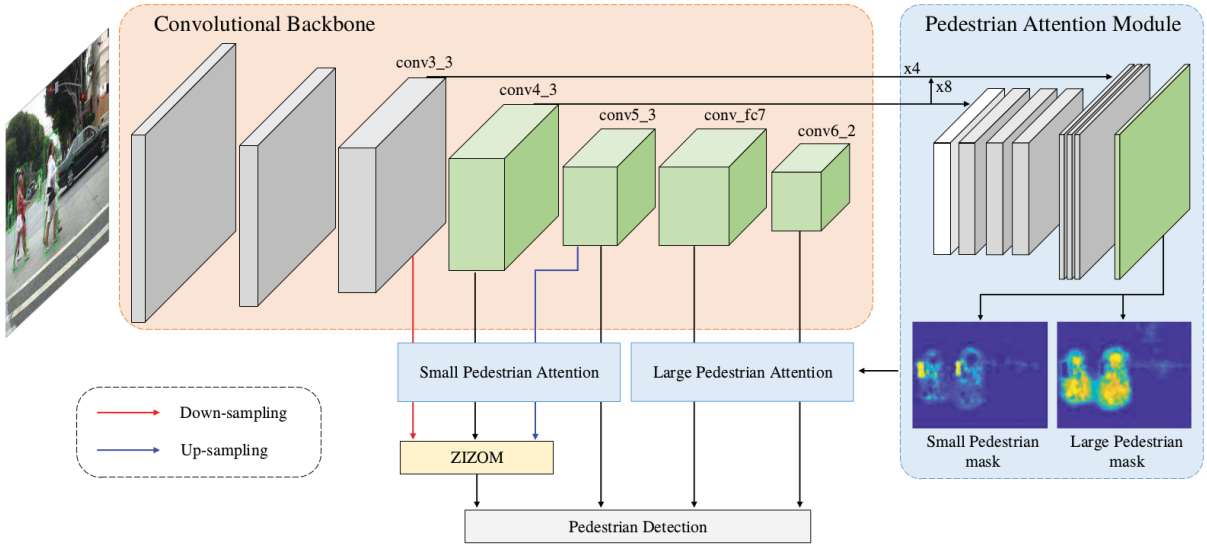


Figure 3.3: GDFL framework composed of three modules: convolutional backbone, pedestrian attention module, zoom-in-zoom-out module. Source: (Lin et al., 2018)

3.2.4 TLL-TFA

Song et al. (2018) tackle the well-known challenge of detecting small-scale in the Object Detection task. They argue that there is a bias in the default type of pedestrians annotations (bounding boxes), which introduce issues in the earliest step in the model training. To prove the existence of bias in the bounding-box annotation style and to demonstrate that there is a better alternative to the annotation, they elaborate a detailed analysis over applied annotations while achieving the state-of-the-art in the Caltech dataset.

They explain that detecting small-scaled pedestrians is a hard task due to appearance challenges such as blurred boundaries and obscurity, which makes arduous to distinguish pedestrians from the background. Additionally, they consider that existing detection methods frequently use bounding boxes as annotations, wherein they include parts of the background that are False Positives corresponding to about 50% of the annotated area. Such unwanted information confuses the detector, where the situation is aggravated for small-scale pedestrian annotations which contains more of such information than ones on a bigger scale. In order to work in the situation, the authors propose to improve the annotation technique.

Song et al. (2018) apply Somatic Topological Line Localization (TLL) to generate an alternative kind of annotation (Figure 3.4). This TLL annotation enables creating 2D Gaussian kernels in different scales as a model of human body shapes. Additionally, the new annotations (TLL) demonstrated to be created more consistently by the annotators when compared to the bounding boxes. They prove that benefit by executing an experiment with multiple annotators where they did label the same objects with both methods. According to the demonstrated results, they achieved higher labeling quality using the TLL annotations.

The authors use a Fully Convolutional Neural Network (FCN) as the basis of the topological line annotation, taking feature representations and regressing the confidence of topological elements. Those elements are related to the annotation being the bottom and top

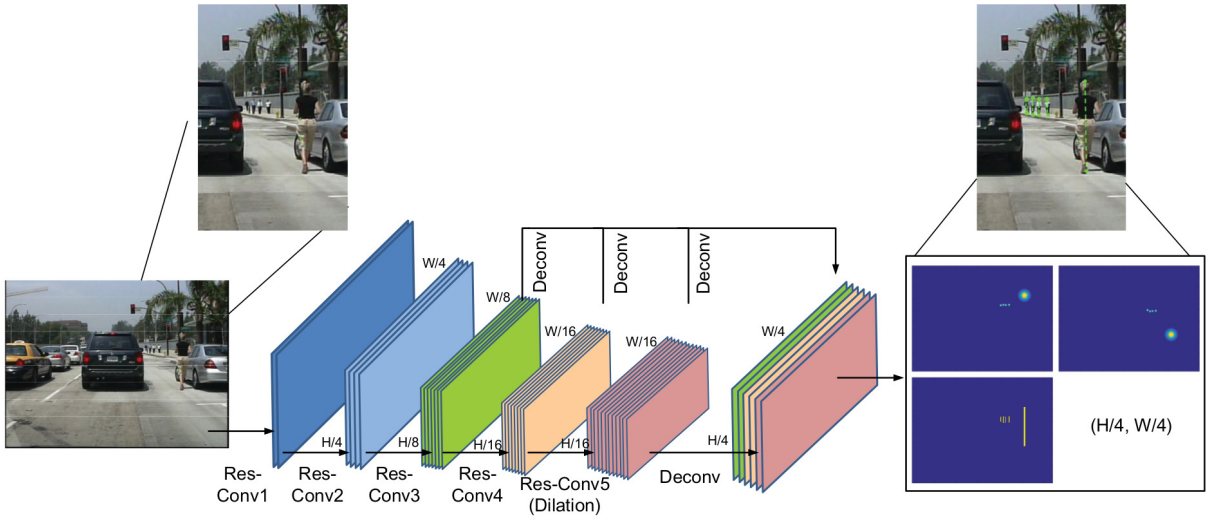


Figure 3.4: TLL Detection Network. In the top-right image, the dotted green lines represent the TLL annotations. In the bottom-right image, the network predicts the top and bottom vertexes, as well the link between them. Source: (Song et al., 2018)

vertex, and their link. The annotated lines are directly ground truths, and they do not generate bounding-boxes. That is why the FCN needs to be prepared to calculate regression that matches this format. Also, it is expected to encounter matching problems in occlusion situations, and to handle that the authors apply the Markov Random Field (MRF) to eliminate ambiguities, which controls the distance between close instance predictions.

Lastly, they aggregate features from adjacent frames with the objective of gathering temporal information used to improve the general performance of the detector. In order to keep the architecture a single pipeline, instead of using the popular Recurrent Neural Network (RNN) they have made use of "Convolutional Long-short Term Memory" (Conv-LSTM) (Xingjian et al., 2015) to propagate features between frames according to the time-flow. The Convolutional Long-short Term Memory (Conv-LSTM) works by sharing convolutional layers and keeping a single state. Every frame processed will update the state and collect the temporal cues as an additional feature for the regression.

The authors evaluate their framework over Caltech (Dollár et al., 2009b), holding the position of the current best-ranked method in the majority of the metric configurations. They also evaluate the method on CityPersons (Zhang et al., 2017) and KITTI (Geiger et al., 2012), achieving top performances. There is no report of detection speed and neither for source-code availability.

3.2.5 SDS-RCNN

Brazil et al. (2017) proposed the exploration of the influence of semantic segmentation to enhance the pedestrian detection quality without impacting in the detection speed. They have introduced a "segmentation infusion network" to simultaneously supervise the learning of pedestrian detection and semantic segmentation. According to the author, such supervision does benefit the pedestrian detection by the indirect improvement of the features generation through shared layers among both tasks. The feature maps became more semantically significant, handling better situations of occlusion and shape variations.

Brazil et al. (2017) have been inspired in their method by the idea of taking complementary advantages of the two main approaches of detecting pedestrians. The first is the object detection

which is efficient in finding the location of multiple instances of the same class, but at the same time returns not many details about the object shape. The second is the semantic segmentation which is effective in determining the objects boundaries but suffers in separating objects of the same class. The authors explore such good qualities to improve the pedestrian detection by proposing to infuse semantic segmentation masks in shared CNNs layers. The infusion helps to put strong cues about the pedestrian location, thus "illuminating" the regions with pedestrians in the feature map. The authors call the technique as "simultaneous detection and segmentation R-CNN" (SDS-RCNN).

They introduce a two-stage framework, composed by RPN+BF (Zhang et al., 2016a) and Faster R-CNN (Ren et al., 2015), which the infusion is applied in both stages (Figure 3.5). In the first stage, the RPN proposes pedestrians candidates, using as backbone the VGG16 (Simonyan and Zisserman, 2014). An infusion layer is attached to the backbone as another "head" which will generate segmentation outputs.

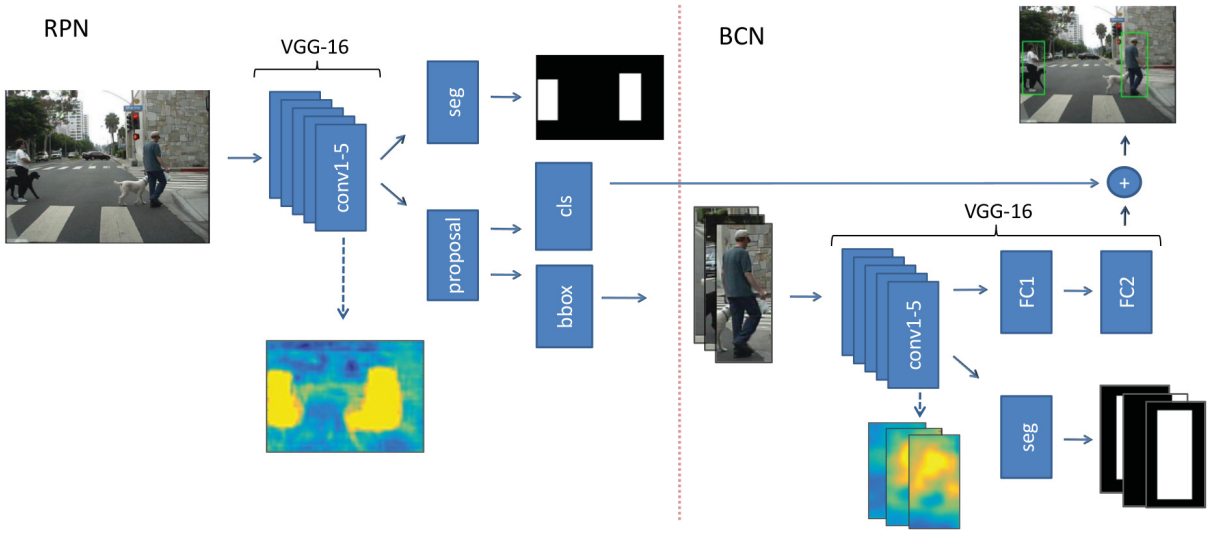


Figure 3.5: SDS-RCNN Framework. RPN in the first stage and BCN in the second. The heat-map figure is a feature map illuminated by the infusion. The black-white figures are the ground truth semantic segmentation masks. Source: (Brazil et al., 2017)

In the second stage, a Binary Classification Network (BCN) will work over the bounding-box proposals generated in the previous stage. Instead of sharing the VGG16 backbone, they choose to create a separated VGG16 network with the argument of avoiding accuracy degradation. Once the first stage already generates the proposals, the second stage aims to work on the "harder" instances by providing a specialized classification using the separated backbone in the Faster R-CNN architecture.

Additionally, a weighting policy is added in order to give more importance to large-scale pedestrian over the small ones. The idea is that closer pedestrians are more urgent to be detected than the ones whose are far away, and the feature for the large-scale pedestrians are more interesting for the small-scale detections.

Another policy is applied to improve the robustness of the predictions. Higher precision is demanded to the classifier, elevating the minimum acceptable threshold from 0.5 to 0.7 of Intersection-over-Union (IoU), that is, a prediction will be considered correct only if the proposal is very precise. This reduces the final number of False Positives by suppressing the-low quality detections. Finally, every proposal is padded in 20% in order to gather more context information, helping to correct dislocated detections as well.

The application of infusion does not degrade the detection speed because the technique is only applied in training, not being necessary for the inference time. That occurs because the wanted effect from the infusion is merely related to the weights enhancement. That is, the infusion will influence the updates in the backbone weights coming from the segmentation task while the detection tasks do the same. Once the training is over, the backbone become able to handle both detection and segmentation. However, just the generation of bounding boxes predictions are desired, and with no losses, the segmentation head of the network can be eliminated.

The authors explain that due to the lack of semantic segmentation annotation in most of the datasets they needed to improvise. They have utilized a technique called "weak semantic segmentation." This technique converts the bounding-box annotations (rectangles) in semantic segmentation masks. At a certain depth of the network, a very precise and detailed segmentation mask will be reduced to rectangle due to the dimensionality reduction. Because of that, a well-shaped segmentation mask of pedestrian and a weakly-shaped will be very similar at some point the network. As the authors apply the infusion not at the beginning of the network, such approach demonstrated to be effective. Figure 3.6 illustrates the concept.

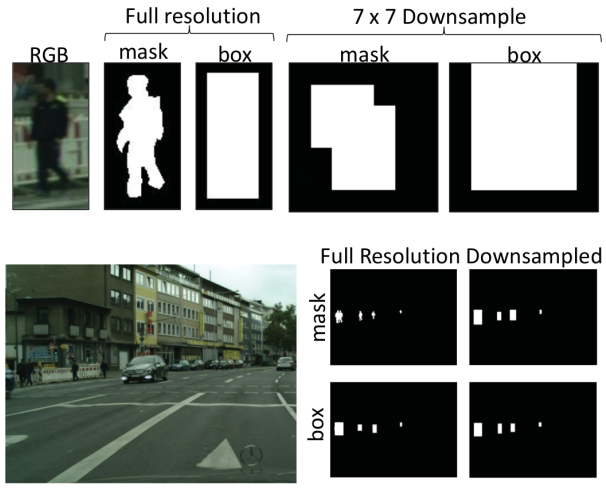


Figure 3.6: SDS-RCNN weak semantic segmentation masks. The images in the bottom are related to the first stage (RPN network) while the images in the top are related to the second stage (BCN network). In the bottom-left, the input image is shown, and in the right, at the first column, it is possible to see the differences among the "box" masks (bounding box based) and the "mask" (default semantic segmentation masks). However, in the second column, the differences are extremely decreased due to the lower resolution provided after certain depth in the network. In such a case, the effectiveness of the masks will be very similar. In the top images the same process is demonstrated but a bit less effective depending of the initial proposal resolution. Source: (Brazil et al., 2017)

The SDS-RCNN has been evaluated over the Caltech and KITTI datasets, achieving top-tier results, including the best accuracy in the main metric configuration. At the time, it was the faster method on the top ranking but now has been overtaken by the GDFL (Lin et al., 2018). The authors publicly release the source code, implemented over the Caffe framework and MATLAB.

4 YOLO: REAL-TIME OBJECT DETECTION

In this chapter, we describe the YOLO object detector. It is the core method in this work, being submitted to many experiments according to the presented proposals. The main characteristics of the detector are reviewed, as well the general architecture and the main differences among its three versions.

YOLO is a real-time object detection system, known by being fast and accurate. It is elaborated as a single-stage detector framework, corresponding to a unified pipeline for the training and detection process, where only one network do all the work. YOLO stands for "you only look once," referring to the fact that it processes the entire image a single time at once, generating the object predictions (Figure 4.1).

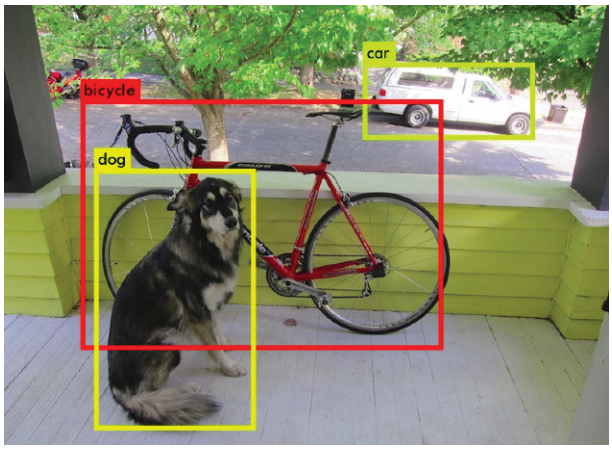


Figure 4.1: YOLO detection examples. Source: https://pjreddie.com/darknet/yolo/.

The YOLO has been created having in mind the applications that require low processing latency, such as autonomous vehicles and robotics. According to Redmon and Farhadi (2017), the YOLOv2 (second version of YOLO) achieves competitive results on the ImageNet challenge (Krizhevsky et al., 2012) requiring only 8.52 billions of operations compared to 30.69 billions of the VGG16 (Simonyan and Zisserman, 2014), which was utilized by direct competitors, at the time, such as Faster R-CNN (Ren et al., 2015). YOLO is fast because it predicts a high number of bounding boxes all at the same time with a single-pass in the network.

YOLO is currently in its third version (Redmon and Farhadi, 2018), but the general idea of all three versions are similar, as follows. YOLO works by dividing the given image in a grid of cells. There is a fixed number of "anchor boxes" for each cell, which corresponds to pre-defined object shapes previously calculated according to the dataset overall objects. In the YOLOv2 (second version), the default grid is 13×13 corresponding to 169 cells, where each cell has five anchors, totalizing 845 possible bounding-box predictions. The bounding box is defined as two coordinates relative to the image matrix corresponding to the central position of the object ("x" and "y"), and the two dimensions of width ("w") and height ("h"). Cells and anchors, in such regions, will predict objects in specific areas of the image. Each bounding box will be accompanied by an "objectness score" that will define how confident the model is about that bounding box containing an object. Also, for each possible object class, there will be one independent probability score, that together must sum 100%. The loss function in the network takes into account the objectness score, the classification of object categories and the regression of the bounding box coordinates/dimensions. The general idea is illustrated in Figure 4.2.

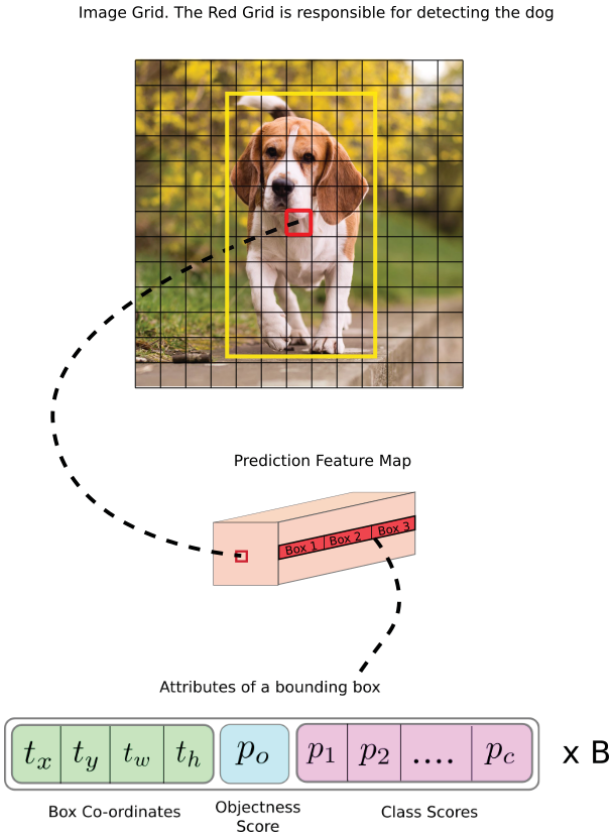


Figure 4.2: YOLO prediction scheme. Source: https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/ by Ayoosh Kathuria.

YOLO filters the predictions by defining a minimum confidence score (threshold) which is defined by default in 30%. Additionally, when training, the "non-maximal suppression" algorithm is applied in order to remove redundant predictions. Predictions that match the same ground-truth object will be compared, and just the ones with the highest confidence are kept.

For each of the three versions of YOLO, the authors also release the respective "YOLO-tiny" variant. This version is much smaller compared to the default ones and focuses in constrained environments. Despite being a shallower network, it achieves good results (half of the default version) while more than 10× faster, that is about 220 FPS. Thus, the YOLO-tiny variants have been frequently used in other applications (e.g., OCR and biometrics) aiming to obtain a better balance between speed and accuracy (Bezerra et al., 2018; Laroca et al., 2019).

All versions of YOLO are implemented with the Darknet framework[1], written in the programming languages C and CUDA, being free and open-source.

4.0.1 YOLOv2 & YOLO9000

At the time, YOLOv2 reached the state-of-the-art on both PASCAL VOC (Everingham et al., 2010) and COCO (Lin et al., 2014) challenges. YOLO did overcome popular and robust methods such as *Faster R-CNN with RestNet* (He et al., 2016) and *SSD* (Liu et al., 2016), presenting competitive results while being faster.

Comparing to its first version (Redmon et al., 2016), the YOLOv2 presents a few improvements:

[1]https://pjreddie.com/darknet/

- Batch normalization is applied to improve the network convergence by providing more stability in the training, and discarding the need for other methods of regularization. This provided an increase of 2% points in the accuracy metric;
- The initial resolution of the network became 448×448 pixels for ten epochs, while in the first version it was 224×224 when training the classifier and later increased to 448×448 for training the detector. Improved the accuracy metric in 4% points;
- YOLO imports the Anchor Boxes technique from Faster R-CNN (Ren et al., 2015). Initially, YOLO predicted the bounding box coordinates directly from the CNNs, but it was demonstrated by Ren et al. (2015) that there was a more efficient way. By using bounding-box priors for each location in the image, the network only needed to predict the offsets for the "anchor boxes," which is an easier task. This slightly reduced accuracy but improved the recall ability.
- Instead of using hand-pricked bounding-box priors (anchor boxes) like Faster R-CNN, in the YOLOv2 they were calculated according to the "k-means" algorithm over the ground-truth annotation from the training dataset. This enabled defining priors that would best fit the general aspect ratio present in the specific dataset, instead of manually defining them.
- The use of anchor boxes created a model instability while training. The offsets were calculated relative to the entire image, being too unconstrained. An anchor box could be utilized in a prediction of a spatially distant bounding box. The authors have found that constraining the location prediction to the local cells made the learning easier and more stable, improving the accuracy in 5% points over the method without the current improvement.
- The new version of YOLO could not handle well small-scale objects. To remedy that, the authors proposed to add a "passthrough layer" which take features from previous layers with a finer-grained feature map. This layer did stack high-resolution features with low-resolution ones in different channels, providing more robust information about small objects. There was an increase of 1% points of accuracy.
- The authors make the model more robust by implementing the support of image of different sizes. To accomplish that, the network input is changed every few iterations, randomly selecting a new image dimension size during the training. The dimensions would vary by multiples of 32, due to the downsampling factor of the network, having as minimum 320×320 and a maximum of 608×608. Such tactic enables better learning of the model for different image resolutions.

Initially, the YOLOv1 (Redmon et al., 2016) network architecture was formed by 24 convolutional layers plus two fully connected ones. The YOLOv2 used a DCNN architecture being fully-convolutional, that is, without fully-connected layers. The network's backbone was called "darknet-19" composed of 19 layers. Eleven additional layers for the specific tasks of detection were added to the backbone, totalizing 30 layers.

The authors also did release a variant to the YOLOv2, named YOLO9000 which was able to detect more than 9 thousand classes. It was simultaneously trained for detection on the COCO dataset (Lin et al., 2014) and classification over the ImageNet (Krizhevsky et al., 2012). According to the authors, due to the combination of the scheme of both datasets, the model can detect classes that are not even annotated in the detection database.

YOLO9000 proposes such a combination of datasets and training methodology due to the lack of data available for the detection task. While the classification datasets are composed of millions of samples, the detection ones contain a few thousands. Acknowledging that scenario, the authors of YOLO9000 wanted to give similar data conditions for the detection training concerning such abundance in classification datasets. However, instead of creating a new large detection dataset, which would be too laborious, they proposed a new method to work around the lack of data by using the already existing image classification datasets. The method uses the hierarchical view of the object tree from the object classification database, and then it is combined with other datasets.

The classification knowledge is acquired by training over ImageNet which contains 1000 classes, while the detection is executed over the COCO with 80 classes. During the training, the network reacts differently according to the origin of the sample. That is, if the image came from the classification dataset the backpropagation is executed only for the classification parts of the network, while the same occurs for images from the detection dataset.

According to the authors, a big challenge was to coherently interconnect the training set of both datasets, to match the object hierarchy tree. In the classification dataset, the tree was more complex, containing a finer hierarchy with more layers, while the detection set they were coarser. For instance, in the detection dataset there was only the "dog" class, but in the classification one the division was made by dog breeds which is a different hierarchy layer. To support such annotation variations, the authors created a multi-label hierarchy model which combines the annotations of both datasets. They called it "WordTree" (Figure 4.3), wherein the combination of both hierarchies resulted in 9418 classes.

4.0.2 YOLOv3

Redmon and Farhadi (2018) released the YOLOv3 with a network composed of 106 layers (Figure 4.4), 53 for the backbone ("darknet-53") and the other 53 for the object detection task, still being a fully-convolutional neural network. Compared to YOLOv2, the authors applied some changes:

- The softmax prediction is replaced by independent logistic classifier with binary cross-entropy loss functions. Such change enabled tackling more complex domains with the multi-label classification, that is, an object can be attached to more than one class. For instance, "tree" and "pine tree."

- Three different scales are now predicted, for small, medium and large scale objects. This approach improves the accuracy of the known weakness of YOLO related to detecting small objects. For each scale, there is one grid: 13×13 for big objects, 26×26 for medium and 52×52 for small.

- Instead of 5 anchors per cell, the quantity is increased to 9, wherein there are 3 for each scale. While YOLOv2 could predict 845 bounding boxes, with this new configuration YOLOv3 can predict 10,647.

- Due to the increase of layers in the network, consequently, YOLOv3 became slower but still being faster than the leading competitors.

- The new network architecture includes "skip connection layers" which introduces similar behavior of Residual Networks.

The YOLOv3 has been evaluated in the COCO Detection Task for two different metrics: mean Average Precision (mAP)-70 corresponding to a stricter accuracy metric and mAP-50 as

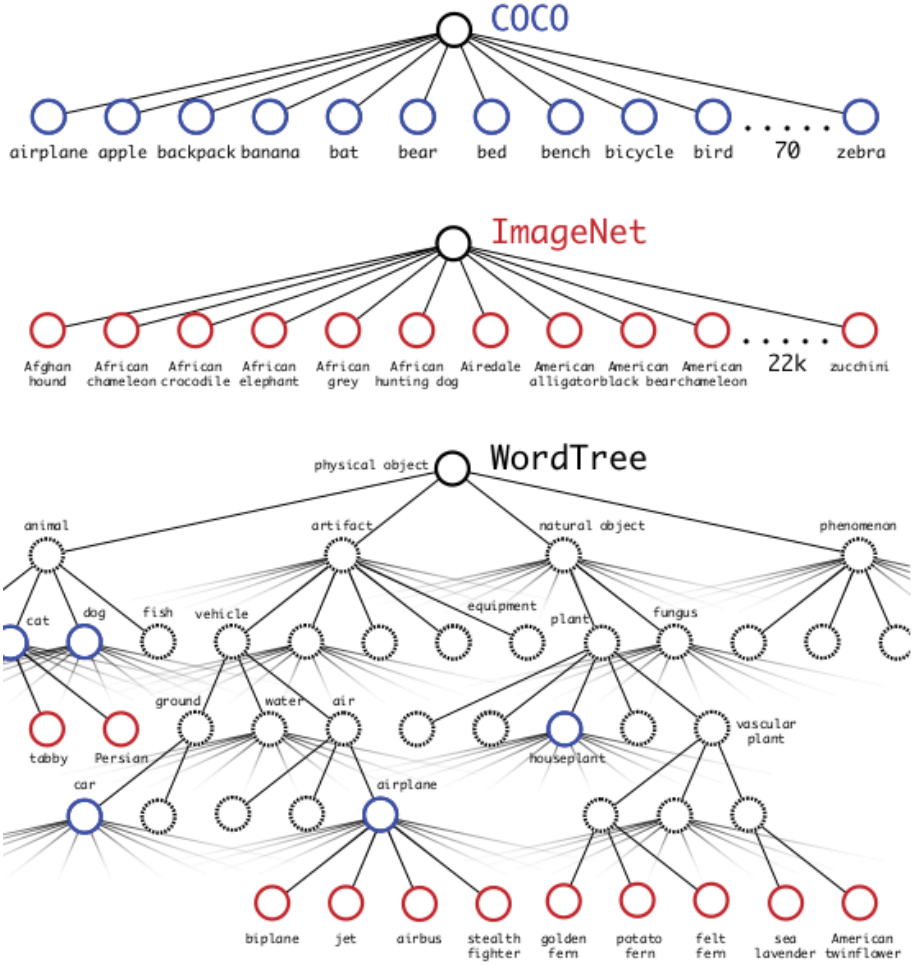


Figure 4.3: YOLO9000 WordTree annotation model, created by the hierarchical combination of class trees from ImageNet and COCO datasets. Source: (Redmon and Farhadi, 2017).

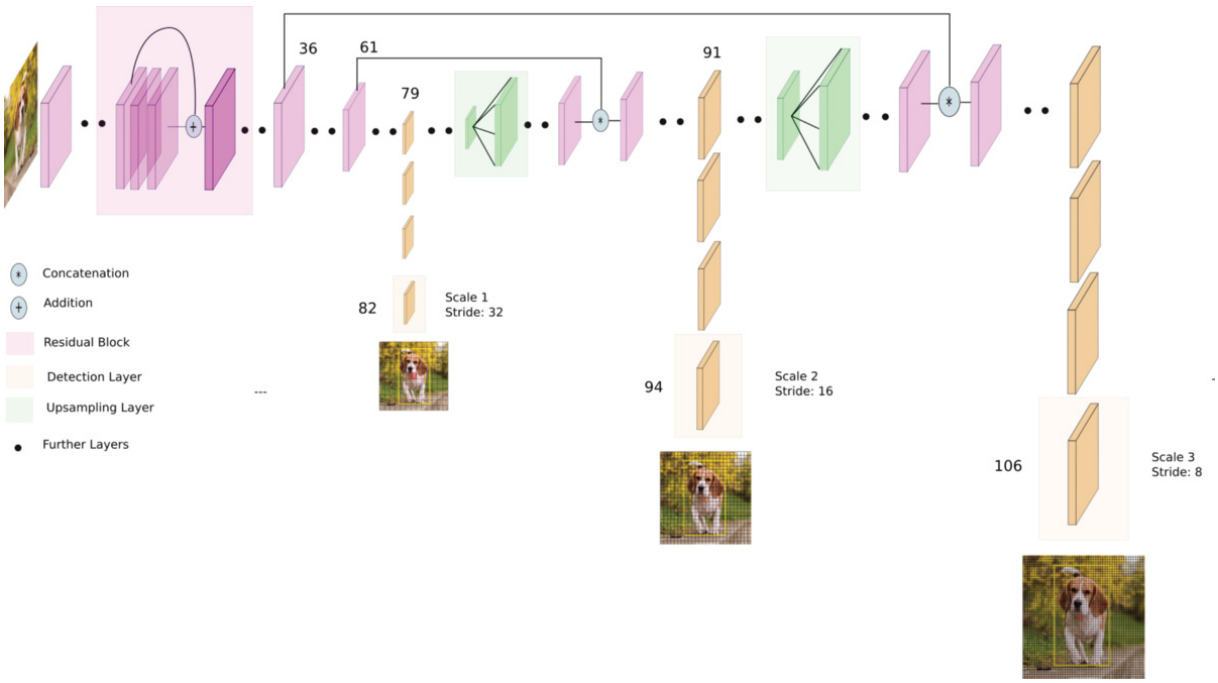


Figure 4.4: YOLOv3 architecture. Adapted from: https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b originally made by Ayoosh Kathuria.

being more tolerant to lower detection qualities which are conventional in the literature. In the benchmark of mAP-50, YOLOv3 reached similar results to the competitors such as RetinaNet (Lin et al., 2017) and DSSD (Fu et al., 2017), while being faster. However, in the benchmark which requires a higher prediction quality (mAP-70), YOLO has a significant decrease in accuracy.

5 PROPOSED APPROACH

We have seen in the literature that Pedestrian Detection is important for real-world applications, such as intelligent video surveillance. At the same time, we also realize that this detection is far from being a solved computational task. While the improvements in the detection are commonly related to the increase of the network parameters, the side-effect is the escalating cost of processing power. As more calculations are necessary to process each image, slower is the detector. We understand that detection speed is essential for many practical applications, for instance, autonomous driving.

Having that stated, this work must offer proposals in such a way that they support the following topics:

- Improving accuracy in Pedestrian Detection is essential, but it is much more significant to increase it without demanding more processing power;
- If there are object detectors that are known by being fast and accurate but haven't been tested yet in some Pedestrian Detection scenario, it would be worth to check its performance compared to the state-of-the-art. Even more, if they report detection speed above the top-tier pedestrian detectors.
- Even if a given object detector is not so accurate as the specialized pedestrians detectors, there may be some techniques that have been already proven to increase the pedestrians detection quality. Consequently, such techniques could be applied to the object detector to check its effectiveness for a different method.

To tackle these topics, we propose to explore the performance of the well-known object detector called YOLOv3 (Redmon and Farhadi, 2018) on the popular *Caltech Pedestrian Detection Benchmark* (Dollár et al., 2012). YOLOv3 is notorious by being fast and accurate, and the Caltech dataset is popular, public and with a solid benchmark and a rich ranking of challenged detectors. As far as the authors of this work know, no works are reporting such evaluation.

We also propose to evaluate the detector on the introduced PTI01 Pedestrian Dataset. This dataset represents a different kind of environment compared to the Caltech dataset. While the Caltech is built for the detection of people from a moving vehicle in an urban scenario, the PTI01 represents a real-world indoor video surveillance network. We find this evaluation important due to the growing enthusiasm demonstrated by the academy in the surveillance area and the interest from the Itaipu Technological Park (PTI) (explained in Section 5.1.3) for practical applications, which the latter could power up the relevance of any achieved results.

As our last proposal, we modify the YOLOv3 neural network architecture by using the *weak semantic segmentation infusion* technique (Brazil et al., 2017). This modification aims to improve accuracy without increasing the processing cost for detection. We analyze the effects on both datasets.

The proposals can be summarized in three approaches which are explained in the next sections:

- Introduction of an additional pedestrian detection dataset for video surveillance experimentation: PTI01;
- Exploration of the YOLOv3 model on Caltech and PTI01 datasets;

- Modification of YOLOv3 neural network to evaluate the effectiveness of an infusion technique that would increase accuracy with no extra detection processing costs;

5.1 PTI01 PEDESTRIAN DATASET

The introduction of this dataset is related to the interest demonstrated by the Enterprise Security Department from the PTI[1]. The authors of this work have interviewed security agents from that department to understand if there were any automation that could help their surveillance work. As expected, they listed some activities which they understand that would benefit from the support from intelligent systems and are related to practical needs in the daily context of PTI. Among the listed applications, we cite at least the following:

- Detection of violence: automatically detect and report on real-time acts of violence, including the presence of weapons;
- Detection of uncommon behaviors: automatically detect and report, for instance, the suspicious flow of crowed in a manner that could reveal some security problem;
- Object Dispatching Detection: automatically detect and report people leaving some object behind, which could refer to losing the object or intentionally dropping it, such as a suspicious backpack left on the floor.
- Theft Detection: automatically detect and report people taking items belonging to others, or that have been unintentionally forgotten in the scene;
- Person Re-identification: scan the video surveillance database and determinate the path taken by a targeted person through the analysis from the imaging of all cameras available, reporting the list of cameras (place) and date and time of each occurrence. That could help to investigate many security cases.

Due to the interest in helping the technological progress in any of the presented opportunities, the PTI's security department has given access to their video surveillance database to the authors of this work.

Despite the existence of some video surveillance datasets[2], we have found the PTI scenario more attractive due to its direct practical case. Training and testing models on this local dataset would express a much more meaningful performance than reporting the results over datasets from a different surveillance environment. Without the presence of negative factors in using the PTI's database, we decided to create the new dataset.

5.1.1 Dataset potential

To build the dataset, as our initial concern, we have chosen the "Person re-identification" (for short, "Person Re-ID") activity reported by the PTI's security department as a useful and supportive technology for daily security tasks. In the current work, the Person Re-ID is not directly approached due to scope limitations. However, according to Zheng et al. (2016a), Person Re-ID is a Computer Vision task composed in a pipeline by at least three specific ones: person detection, tracking, and identification. Also, they state that such sub-tasks are impracticable to be executed by human labor. The automation of those activities is desired once humans could

[1] www.pti.org.br

[2] http://www.cvpapers.com/datasets.html

assume more creative tasks beyond the monotonous action of detection, for example. Zheng et al. (2016a) affirm that in this task's pipeline, the quality of the person detection executed in the first step has indirect influence in the next ones.

Instead of tackling the entire Person Re-ID framework, in this work we chose to improve Pedestrian detection. Nevertheless, due to the approach utilized to build the dataset, it is possible to apply future upgrades enabling it for Person Re-ID evaluations. These modifications would include the addition of tracking and identity information.

5.1.2 Building the dataset

As stated, the proposed dataset will support the Pedestrian Detection evaluation, but will also provide some characteristics to enable future works with Person Re-identification. One of these characteristics is related to selecting cameras which have an intersection in their line of sight. It is similar to the surveillance scenario demonstrated in Figure 5.1. In that way, among the provided images PTI01 includes scenes with recordings of the same people by different cameras at the same time, as can be verified in Figure 5.2.

Additionally, some cameras do not have an intersection. For those cameras, some of the selected frame sequences correspond to the accompaniment of the same people over different scenarios. Lastly, there are random events of other cameras.

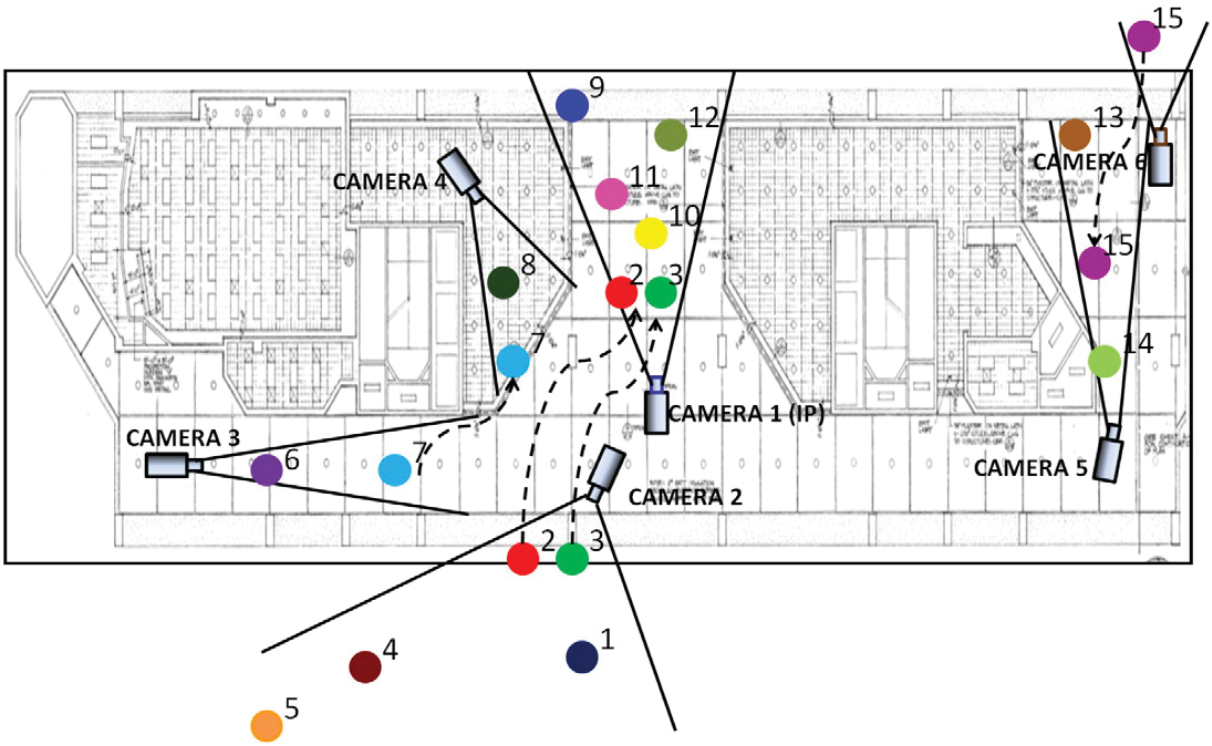


Figure 5.1: Person Re-identification example scenario. The Person Re-ID will, for instance, find the occurrences of the "person number 2" (in the red dots) which will be sequentially in sight of the cameras 2, 3 and 1, according to the person movements in the scenario. In other words, the Person Re-ID will help in finding the path a person took in the environment. Source: (Bedagkar-Gala and Shah, 2014).

The cameras and events selection has been supported by the experience of the security agents from PTI, which participated in the image extraction. For sure, that increased the significance of results obtained with experiments by the usage of more relevant scenes.

The number of cameras was not such a strict objective. For that parameters, we established the goal of having at least more cameras than the well known Person Re-ID datasets,

Figure 5.2: Example of intersection in the points of view of different cameras in the PTI01 dataset. The four cameras capture the same event, that is, the same people in the same place at the same time.

such as PRW Dataset (Person Re-identification in the Wild) (Zheng et al., 2016b) with six cameras. Fortunately, while collecting the frame events, we have reached 21 cameras.

There is another critical parameter when building a Person Re-identification dataset: people's identities. Here the identities are referring to an arbitrary numeric ID that will define that two or more pedestrians are the same, and just that. However, in this work, that information has not been compiled and will not be reported, once the focus is currently strict to Pedestrian Detection.

To choose the number of frames that would be annotated, we defined as a goal to collect the biggest number the current work's scope would support. In respect to that, the authors have been able to prepare almost 8,000 frames and more than 30,000 annotated pedestrians. Comparing to the PRW dataset, they have collected almost 12,000 frames and more than 34,000 annotations. For the Caltech Pedestrian Dataset (Dollár et al., 2009b), its authors have collected about 250,000 frames and 350,000 pedestrian annotations. We conclude that our proposal, regarding the number of cameras, frames and annotations, is feasible comparable to the PRW dataset, suitable for evaluations in Pedestrian Detection even that it is much smaller than the Caltech.

The dataset was annotated in three phases. The first phase focused in a quickly releasing an initial version of the annotations, without much care to the labeling quality. For that version, part of the annotations has been made by the authors of this work and the other part by an online AI platform called "Hive"[3]. The platform offers, among many works, the labeling of datasets

[3]https://thehive.ai/

for a price. However, they provide an initial bonus for trying the platform. We have used that bonus and also configured the parameter of the annotation quality available to the lowest possible, keeping the project within the budget. As expected, due to the low investment the annotations, have been delivered with many wrongs, imprecise or incoherent annotations.

In a second phase, the authors and a few volunteers have re-factored the majority of the low-quality annotations. In order to evaluate the effect of the annotation's quality, we proposed to train and test the pedestrian detector model using the different versions of the dataset.

Lastly, in the final phase and with the volunteers helping again, multiple classes have been introduced for the annotated pedestrians, as well more improvements in the labeling. The authors have proposed these classes through analysis of the dataset that generated intuitions, and by inspirations from Dollár et al. (2012) and Liu et al. (2018).

Liu et al. (2018) state that intraclass variations influence the detection accuracy. Instances from an object class can suffer from severe variations concerning the appearance, including the typical "human" class. There are deformations, variations in pose and even in clothing. The environment and image conditions also affect the appearance of the object, for instance, by variations in illumination, weather, cameras, background, viewpoints, and others. For Dollár et al. (2012), occlusion is another situation that has a significant impact on the detector's performance.

Inspired by those facts, we have observed interesting variations for the pedestrian appearances in the proposed dataset, as well a significant quantity of occlusion occurrences. Therefore, we wanted to measure the impact of such appearance variations according to the model performance. Besides that, we have got interested in evaluating the behavior of the model when isolating, merging or removing some classes.

Eight classes have been elaborated through some investigation, and defined after a consensus about the essential intraclass variations (Figure 5.3), described as follow:

1. "Normal occlusion": more than 50% of the body is visible, including head and majority of the upper body;

2. "Severe occlusion": too much information about the pedestrian is lost. Upper body or head are missing. It is hard to distinguish the shape of the pedestrian from the environment;

3. "Head": only the person's head is visible;

4. "Body part": corresponds to body extremities that are isolated in the scene, for instance, feet, hands, and single leg. This class does not include head;

5. "Top view": these are people viewed from a top angle, that is, they are well below the camera, and they lose the default pedestrian shape. This situation normally occurs in the bottom part of the image. There is very little information about the pedestrian sides or anterior/posterior view;

6. "Far": pedestrians that are interpreted as being too far from the camera. It does not have an exact quantity of pixels because in each camera it would vary. As well, the height of a bounding-boxes does not define a pedestrian to being far away from the camera, because occluded pedestrians sometimes match the criteria;

7. "Angled": these are pedestrians that have their vertical axis changed from about 45 to 90 degrees. Such occurrence is related to positioning of the camera view according to the terrain, wherein some cameras do not contain pedestrians in such situation, and others do;

8. “No occlusion”: does not fit in any other class, where the person has the entire body visible;

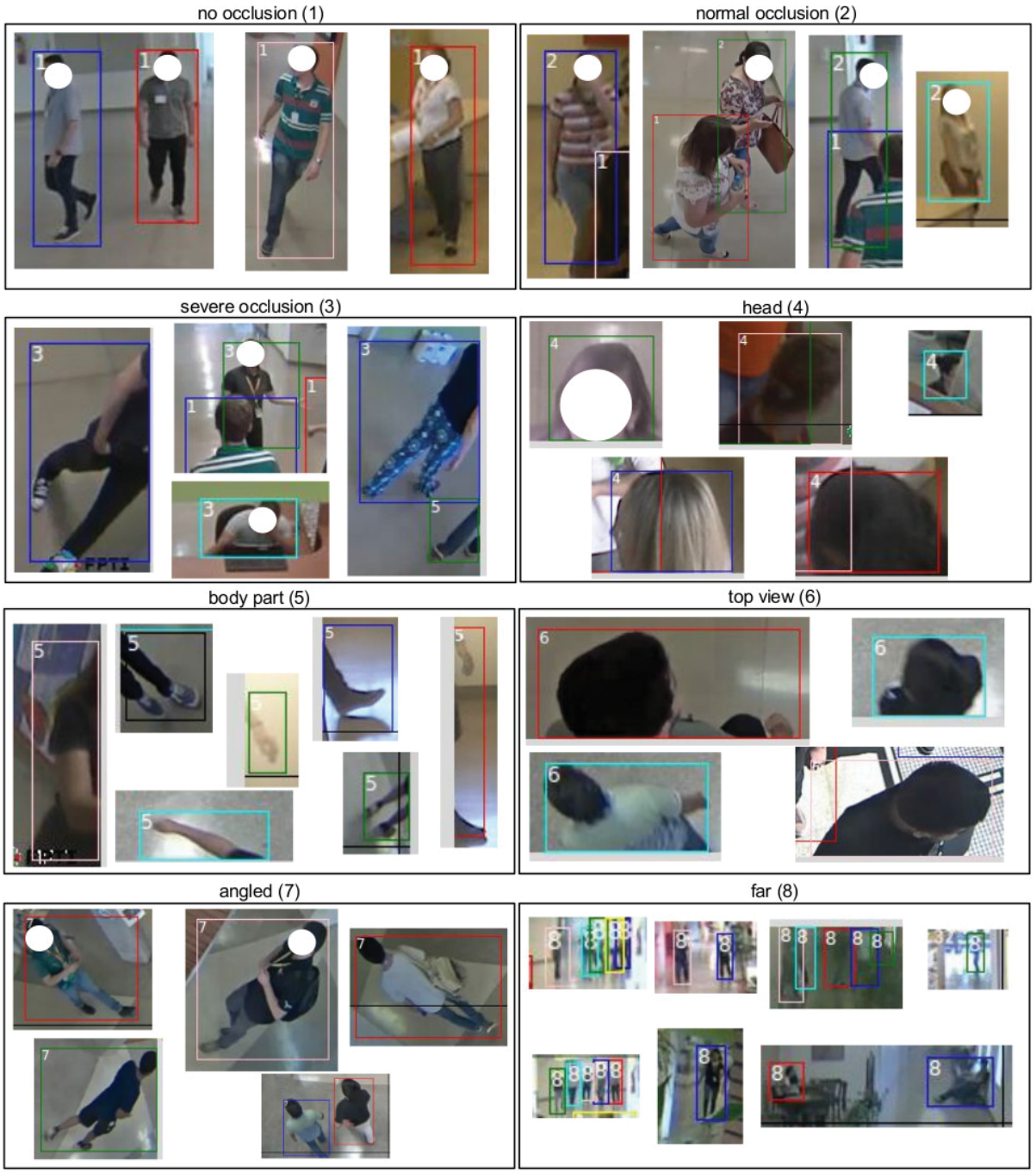


Figure 5.3: Examples of annotations of each class.

The annotators defined the new classes for the labeled pedestrians using their interpretation, which can vary between the annotators. In order to help in the annotation process, two actions have been taken before initiating it:

1. Each camera view has been analyzed, and it was obvious that for each camera there were regions in the image corresponding to a specific class. For example, for a given camera every annotation in the top 50 pixels would correspond in the majority to the class “far,” and in the bottom right, almost all the pedestrians would belong to the “angled”

class. Observing that patterns could be established for each camera, a script has been developed to automatically define the class of an existing bounding-box according to a manual mapping of regions and classes (Figure 5.4). The annotator would need only to review these automatic-labeled bounding-boxes. We believe to have avoided many hours of work by taking this action;

2. We have improved an existing labeling tool[4] to support multi-class annotation in a more agile graphical interface.

Figure 5.4: Examples of regions defined in each camera that enabled the automatic placement of initial classes for the last version of the PTI01's annotations. Pedestrian bounding boxes with the IoU equal or higher than 80% in any of the pre-defined regions would be attached to the correspondent class. Rectangles in green represent the region for "angled" class, while the blue is for the "far" and the yellow for "top-view."

In a final step, the annotations have been reviewed by more than one annotator, and major corrections were applied.

The introduced dataset have been named "PTI01 Pedestrian Dataset," where the numeric value "01" provides a smooth and clear nomenclature for the next possible versions of the dataset.

5.1.3 The PTI and its video surveillance network

According to PTI's official website[5]:

[4]https://github.com/gustavovaliati/BBox-Label-Tool
[5]https://www.pti.org.br/en

> "Itaipu Technological Park (PTI) was created by Itaipu as a product of the power plant's expansion, functioning as the right arm to transform the region through research and development of sustainable technologies."

The PTI is an institution that receives up to six thousand people per day which are registered to visit, work or study in the park. As a resource of enterprise security, PTI has a video surveillance network composed of more than 250 cameras. The security agents, daily, monitor the park through the available real-time imaging, as well the recorded video database for many security analysis activities.

The ZoneMinder[6], an open-source video surveillance software system, supports the security department in the monitoring task, providing the real-time visualizations as well managing the image recordings. The system and the data are hosted in dedicated servers and network infrastructure. The camera images are persisted directly in frames with dimensions of 640×480 pixels in roughly ~2.5 FPS, varying from around 0.8 to 5 FPS depending on the systems' load. The frame rate is not a precise value due to variations in many components of the system that influences the general "Input/Output." Among the different types of recording models supported by the ZoneMinder, the PTI's Security Department makes use of two of them. The first is the "RECORD" mode in which the images are continuously recorded and persistent in events of fixed length. The second is the "MODECT," where the recordings are oriented to scene events which are equivalent to moments where movements are detected in the scenario. In that way, only scenes that contain some action are persisted, saving storage resources and reducing the amount of non-significant data.

To give an interesting idea about the PTI's video surveillance network dimensions, we present Table 5.1 which demonstrates how much data could be generated in the system at ~2.5 FPS, 250 cameras where each frame would have about 50 KBytes. This table presents the statistics provided by the PTI's security department, revealing the amount of data stored daily in the two modes of recording. As it can be verified, when using the mode of motion detection the amount of data is dramatically reduced to approximately $\frac{1}{5}$. The statistics for seconds, minutes and hours corresponded to average calculations based in the total amount of Megabytes recorded in an entire day. Because of that, the quantity of 124 frames recorded per second is just a mean. There will be moments in which cameras are recording and moments when they are not. With the Events Control enabled, the FPS of an active recording is still ~2.5. If a pedestrian detection system wanted to process the entire video monitoring network, that would be the minimum amount of data it should be able to process for each active recording camera, at a maximum of 625 FPS (250 cameras simultaneously).

Table 5.1: PTI's video surveillance network dimensions. The "Continuous Recording" column presents the statistics for 250 cameras recording nonstop. The "With Events Control" corresponds to the recordings when the Motion Detection is enabled. The calculation of events control is based on the number of Megabytes in a day.

	Continuous Recording ("RECORD")		**Motion Detection ("MODECT")**	
Time slice	**Size (MBytes)**	**Frames**	**Size (MBytes)**	**Frames**
1 second	10.11	207.20	6.07	124.42
1 minute	606.79	12,431.87	364.37	7,465.21
1 hour	36,407.47	745,912.42	21,862.29	447,912.38
1 day	873,779.20	17,901,898.06	524,695.00	10,749,897.00
1 month	26,213,376.00	537,056,941.69	15,740,850.00	322,496,910.00

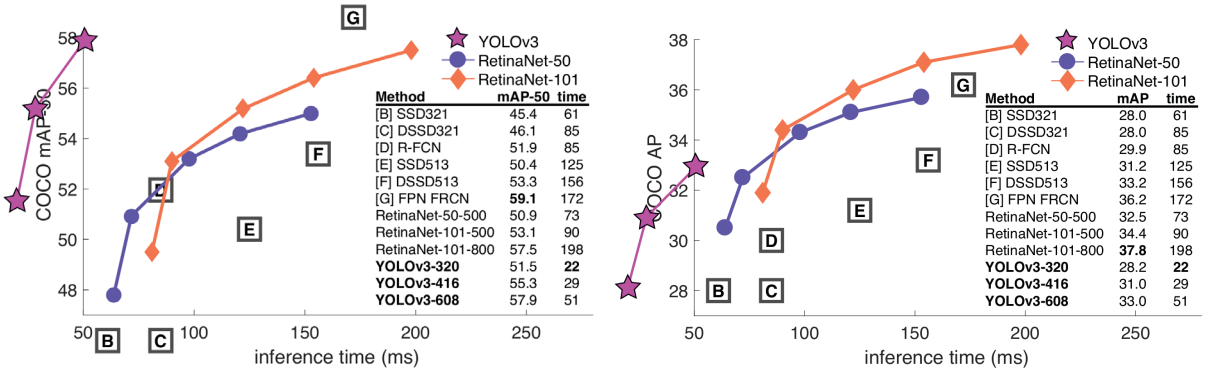
[6]https://zoneminder.com/

5.2 YOLOV3 EXPLORATION

The proposal of experimenting YOLOv3 (Redmon and Farhadi, 2018) on the Caltech Pedestrian Dataset (Dollár et al., 2009b) arises from the analysis of the main results reported in the Caltech and COCO Detection Task (COCO Dataset) (Lin et al., 2014). Both datasets sustain stable and well-known benchmarks, one for Pedestrian (Caltech) and the other for Generic Object Detection.

It is possible to observe in the Caltech benchmark that, for the specific *reasonable metric*, Brazil et al. (2017) holds the challenge's leadership with 7% of miss rate together with Song et al. (2018). The former work reports a detection speed of ~0.21 seconds for each image (~4.8 FPS.), while the latter does not report the speed. Just for comparison, Du et al. (2018) also achieve high accuracy (8% miss rate) by a cost of ~2.5 seconds for detecting a single image/frame (0.4 FPS). The fact that Song et al. (2018) did not report speed is curious, probably meaning that it is so slow that is not worth to be reported, or despite being competitive, the authors have chosen to not report it. Such results demonstrate that the detection speed varies among the top-tier works.

Changing to the COCO Challenge, we take into account two of the most accurate works (Figure 5.5). For the metric mAP-70 (explained in Section 6.2), Lin et al. (2017) using the *RetinaNet-101-800* method, achieve the best accuracy requiring 198 ms to process an image. To increase the speed performance, RetinaNet can downgrade its core neural network structure (backbone), reaching to 70 ms but giving up of considerable accuracy. Among other competitors such as FPN FRCN (Lin et al., 2016) and DSSD (Fu et al., 2017), we highlight the *YOLOv3* object detector made by Redmon and Farhadi (2018). They achieve competitive accuracy at the cost of 51 ms per image, or a lower accuracy for 22 ms of detection time. Changing the metric to mAP-50, which has been the more conventional one in the literature, YOLOv3 passes RetinaNet regarding accuracy, while being 4× faster.



(a) mAP-50 (b) mAP-70

Figure 5.5: Performance of top works in COCO for both 50% and 70% IoU threshold in mAP. YOLOv3 does better than RetinaNet in the mAP-50 which is the more conventional metric and still is competitive in the mAP-70. However, YOLOv3 is about 4× faster than RetinaNet. Source: from (Redmon and Farhadi, 2018) initially adapted from (Lin et al., 2017)

It was possible to understand from the literature that detection speed is so relevant as accuracy. With that in mind, YOLOv3 takes attention for such speed in detection and being competitive in accuracy. However, curiously, we have not seen YOLOv3 in the official Caltech Pedestrian Benchmark reporting list. That is, by such report or any other means, it is unknown to us the evaluation of YOLOv3 in such well-known Pedestrian Detection scenario. Because of this fact, we see the application of the detector in the Caltech challenge as an opportunity of exploration. We call it "exploration" because we propose a few configurations while evaluating the detector, and we report them as follows.

Instead of directly exploring the YOLOv3 on the Caltech dataset, we define the PTI01 as our initial experimental dataset (more discussed in Section 6.5.1). We have concluded that it would not be worth to use Caltech in the early experiments once it is extremely time-consuming for training and because it is a much harder scenario compared to the PTI01 dataset. We choose to elaborate some evaluations on PTI01 primarily, and after some conclusions, we reproduce the more relevant ones and to apply a lower range of experimental variations on the Caltech dataset.

Dollár et al. (2009b) proposed to standardize the width of both ground-truth and predicted bounding boxes to avoid undesirable effects by variations in that dimension. They modify the bounding boxes width so that they match a pre-defined and fixed aspect ratio which was calculated as the mean of all ratios in the Caltech's dataset. With this technique, they report positive effects when testing detectors which usually return narrow bounding boxes. Dollár et al. (2012), also report a degradation in accuracy when detecting bounding boxes with atypical aspect ratios, that is, the extreme variations in the pedestrian shape decrease the detection quality. Due to what has been stated, we consider that working with the aspect ratios could bring some improvements in the YOLOv3's performance.

Inspired by Dollár et al. (2009b), we want to experiment with a simple mechanism of "standardizing aspect ratios" by the creation of what we call "canonical bounding-boxes." As stated by (Dollár et al., 2009b), the benefit of such an approach was reported for detectors which tended to generate narrow bounding boxes. As this is not the case for the YOLOv3, we feel not obligated to reproduce the same configuration in our experiments. Instead, we propose to adjust the PTI01's bounding box dimensions ensuring that the relation between height and width would always be matched to pre-defined ratios. This will produce multiple standardized aspect ratios and not a single one. Figure 5.6 demonstrates that the majority of the bounding boxes correspond to aspect ratios (height/width) between 1.8 and 3.4, and the overall range goes from 0.1 to 4.9. Observing the aspect ratio distribution, we have defined three ratios: 1.0, 2.0 and 3.0. In our opinion, these values would cover the majority of the bounding boxes. We do not consider using the aspect ratio of 4.0, because it is visually too discrepant for the general aspect of the pedestrian shape, being too narrow and tall. The shape of 1.0 would fit in the similar visual discrepancy, but it was kept because it normally represents pedestrians in a "non-full-body" view. The overall objective is to understand whether multiple visually coherent standardized aspect ratios would improve the model's performance.

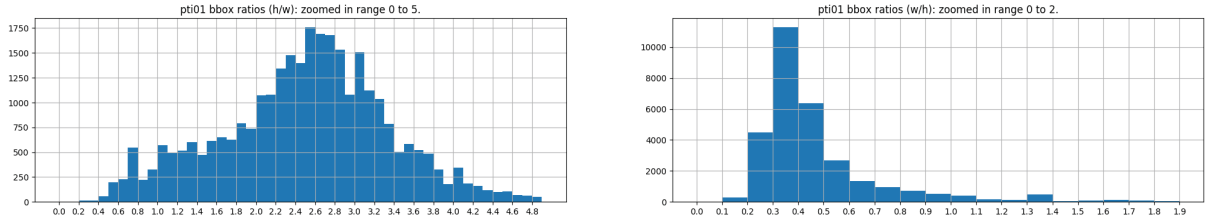


Figure 5.6: PTI01 general aspect ratio. On the left height/width. In the right width/height.

There is a default procedure when testing YOLOv3 in a new dataset, which is recalculating its "anchors." We want to compare the results with different configurations in anchors: 1) calculated for the COCO challenge (Lin et al., 2014); 2) recalculated for the PTI01 Dataset; 3) recalculated with simple k-means clustering algorithm, using a different number of anchors (clusters).

To understand the impact of intra-class variations, explained in Section 5.1, we propose to explore the impact in varying the class combinations while training. That enables the measurement of how the changes in appearances of pedestrians are reflected in the model's ability

of learning and predicting in such cases. It will also be possible to check the impact of occlusions reported by Dollár et al. (2012).

Liu et al. (2018) state that "data augmentation" is commonly used in the literature in order to achieve more robust feature representations in the CNNs. This data augmentation synthetically modifies the training images while keeping the object representation still valid. These increase the object appearance variation by tricking the model as if there would be a higher quantity of different images for training. YOLOv3 does implement data augmentation by default in the training process, and we find it relevant to check the impact of it regarding accuracy and training time for a Pedestrian Dataset.

While the YOLOv3 has been projected initially as a generic object detector, being able to detect more than 9,000 classes (Redmon and Farhadi, 2017) or 80 in the COCO Challenge (Lin et al., 2014), we suppose that when training with one (pedestrian) or just a few classes (e.g., eight classes) it would be possible to deactivate the data augmentation or even decreasing its intensity. The YOLO's default data augmentation algorithm applies the following procedures: randomly resize the network input; randomly crop part of the image; randomly changes color features like hue, saturation, and exposure; randomly decides to horizontally flip the image; randomly resize the dimension proportions (width and height). Restricting the augmentation procedures could provide a faster training time with similar accuracy.

YOLOv3 (Redmon and Farhadi, 2018) has achieved outstanding results in the COCO Object Detection Task (Lin et al., 2014). Such results are achieved with their default model. However, there are other versions of the YOLO model, for instance, the YOLOv3-tiny. This is not reported in the paper but is made available[7] by the author. The YOLOv3-tiny is a much smaller version of the default YOLOv3, created with the objective of consuming much less processing power to run while having impressive accuracy. Because of our interest in detection speed and consistent detection quality, we propose to also experiment in this smaller version of the default model.

Lastly, we want to compare two different configurations of the Caltech Dataset annotations. According to Zhang et al. (2016b), the Caltech dataset suffers from some bad quality annotations (more details in Section 6.4). Because of that, they have proposed a reviewed version of those annotations which we find interesting to evaluate with the YOLOv3 detector. In that way, we can analyze the influence of such low-quality annotations when compared to better ones and then verifying the impact in the detector's performance.

5.3 YOLOV3 + INFUSION

The YOLOv3 (Redmon and Farhadi, 2018) is a generic object detector, and because of that one should not expect it to be directly comparable to the state-of-the-art in a Pedestrian Detection scenario. Even so, one should also not underestimate the generalization capabilities brought by a model able to efficiently detecting more than 80 different classes on a famous and challenging competition (COCO (Lin et al., 2014)).

To understand the natural performance of YOLOv3 as a pedestrian detector, in Section 5.2, we propose the exploration of the model and the comparison to other methods which hold the top accuracy in the Caltech Pedestrian Detection Benchmark. Additionally, it is also evaluated on the PTI01 Pedestrian Detection dataset, which gives an interesting grounding for the YOLOv3's performance on a well-known video surveillance scenario to the authors of this work.

The YOLOv3 is not a final model and has possibilities for improvements. That can be proved by the own historical evolution of the YOLO (Chapter 4), wherein it has been

[7] https://pjreddie.com/darknet/yolo/

continuously improved for three versions which explored the adoption of recent techniques that have demonstrated to be successful in the literature. In such way, each new version of YOLO has been aggregating functionalities that improve its detection quality, and there is no signal of model's saturation.

One of the most notable characteristics of YOLO is its detection speed. In the COCO competition, no other works achieve greater accuracy than YOLOv3 at velocity it can detect objects in images. Other works like RetinaNet (Lin et al., 2017) present a slightly higher accuracy but at the cost of being four times slower than YOLO. In such a scenario, when introducing new techniques to the YOLOv3 algorithm, the decrease of the detection speed should be carefully considered.

In order to explore the enormous potential that YOLOv3 has demonstrated as an object detector, beyond testing its natural performance as a pedestrian detector we want to try to improve that ability. In the same way as YOLO has been adopting strategies from other methods to increase its detection quality, we want to improve the pedestrian detection through the application of a technique utilized by a successful work in the Caltech's benchmark. The technique is called "weak semantic segmentation infusion" and is applied by the SDS-RCNN (Brazil et al., 2017) which achieved top accuracy in the Caltech's "reasonable" metric. According to the authors, the infusion provided a reduction of about 3% in the Log-Average Miss Rate (LAMR) metric, where the lower, the better. This technique is also interesting due to the fact of not inflicting additional processing costs at the inference time. The infusion is applied only in the training phase and removed while predicting. This technique fits well for our purposes of trying to improve the YOLOv3's model accuracy for pedestrian detection and keeping its default detection speed.

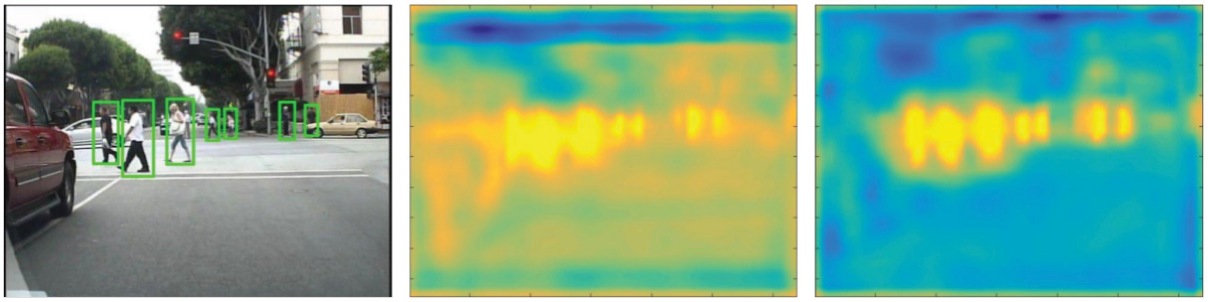
5.3.1 Understanding the application of the infusion technique

In this Section, the general ideas of the proposed application of the "weak semantic segmentation infusion" in the YOLOv3 model are explained. The review of the SDS-RCNN (Brazil et al., 2017) and the general ideia of technique have been already introduced in Section 3.2.5.

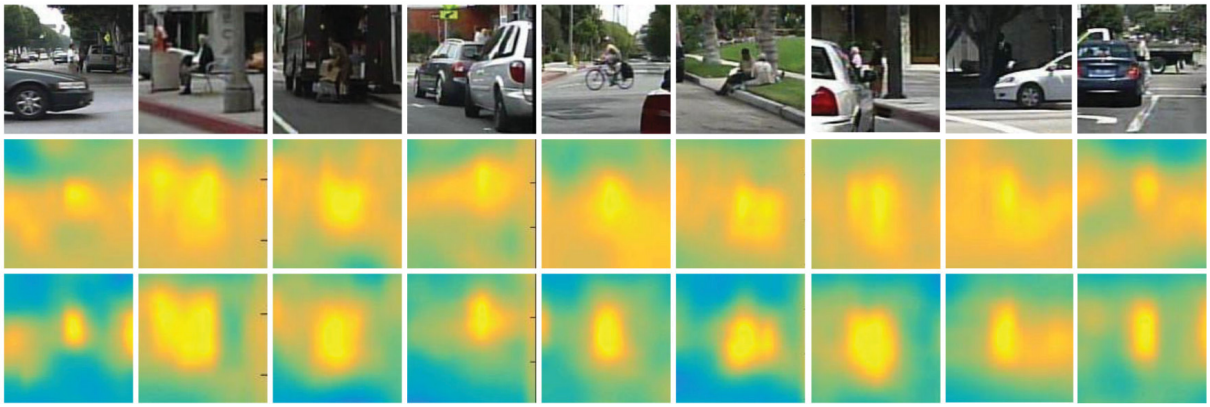
We propose to apply the infusion similarly as Brazil et al. (2017) did. They have created an additional prediction head connected to the end of the model's backbones (Figure 3.5). That head would predict only semantic segmentation masks, marking every pixel as background or pedestrian. Such structure obligates the network's backbone to acquire enough knowledge to, at the same time, predict both bounding boxes as well the masks. The desired effect is that the backbone gets smarter through the additional learning required which will, according to Brazil et al. (2017), enhance the probable locations of pedestrians in the image through the semantic segmentation clues (Figure 5.7). This will improve the bounding box prediction training because the segmentation head suppresses areas with no pedestrians and the ones with pedestrians are "highlighted." The backbone is updated to understand such regions in each image, and the bounding boxes predictions start to get more focused on pedestrians locations through the clues given by the segmentation task.

The SDS-RCNN backbone is a modified VGG16 network (Simonyan and Zisserman, 2014), composed of only five convolutional layers. The YOLOv3's backbone is called "darknet-53" and is composed of 53 layers, being that 38 are convolutional. This gives an idea of how different both methods are when compared to each other. In this experimentation due to the enormous differences, the final results may be different, and the infusion technique could not be useful as it was in the SDS-RCNN.

The first step to include the infusion technique in the YOLOv3's architecture is to create the additional head for the semantic segmentation predictions. The YOLOv3 has by default three heads, each one predicting in different resolutions (Figure 4.4), while the YOLOv3-tiny



(a) Left: final pedestrians predictions. Middle: feature map without infusion. Right: feature map with the infusion.



(b) Top: original scene. Middle: feature map without infusion. Bottom: feature map with infusion.

Figure 5.7: Figures "a" and "b" demonstrate examples of the effect of the infusion technique in the feature map visualizations of the Region Proposal Network (RPN) utilized by the SDS-RCNN. With infusion, the background is suppressed, and the attention (hot colors) is more concentrated in the pedestrian locations. Source: Brazil et al. (2017).

model works with only two. To create the fourth head for the YOLOv3 model and the third for the YOLOv3-tiny, we inspect the SDS-RCNN implementation. It is composed of one convolutional layer with only two filters, one for background and the other for pedestrians. After the convolution, the "batch normalization" is applied to normalize the output. The YOLOv3 applies this normalization after every convolutional layer. The final output is generated by the "softmax" activation function that will calculate the probabilities of each pixel in the image being background or pedestrian, according to the two filters previously implemented in the convolutional layer. In that way, for each image inserted in the network, the semantic segmentation head will produce two images. The first will present the pixel values closer to 1.0 when a pixel may be referring to a pedestrian, or a value tending to zero when it may refer to the background. The second image does the inverse. For instance, if a pixel has a value of 0.8 in the first image, that means it is 80% confident that the pixel belongs to a pedestrian. While in the second output, the same pixel will have a value of 20% but now presenting the confidence of the pixel being related to the background.

The implementation of the infusion technique is also elaborated for the YOLOv3-tiny model. It follows the same principles for the default model, having as main difference a much smaller network size. The YOLOv3-tiny backbone has only eight convolutional layers which are closer to the VGG16 network size. Figure 5.8 demonstrates YOLOv3-tiny architecture diagram including the infusion head. The diagram zooms in the lower part of the network, not displaying

the backbone to improve the visualization. In the same way, we have chosen not to display the YOLOv3 diagram because it is too big to fit in figure visually.

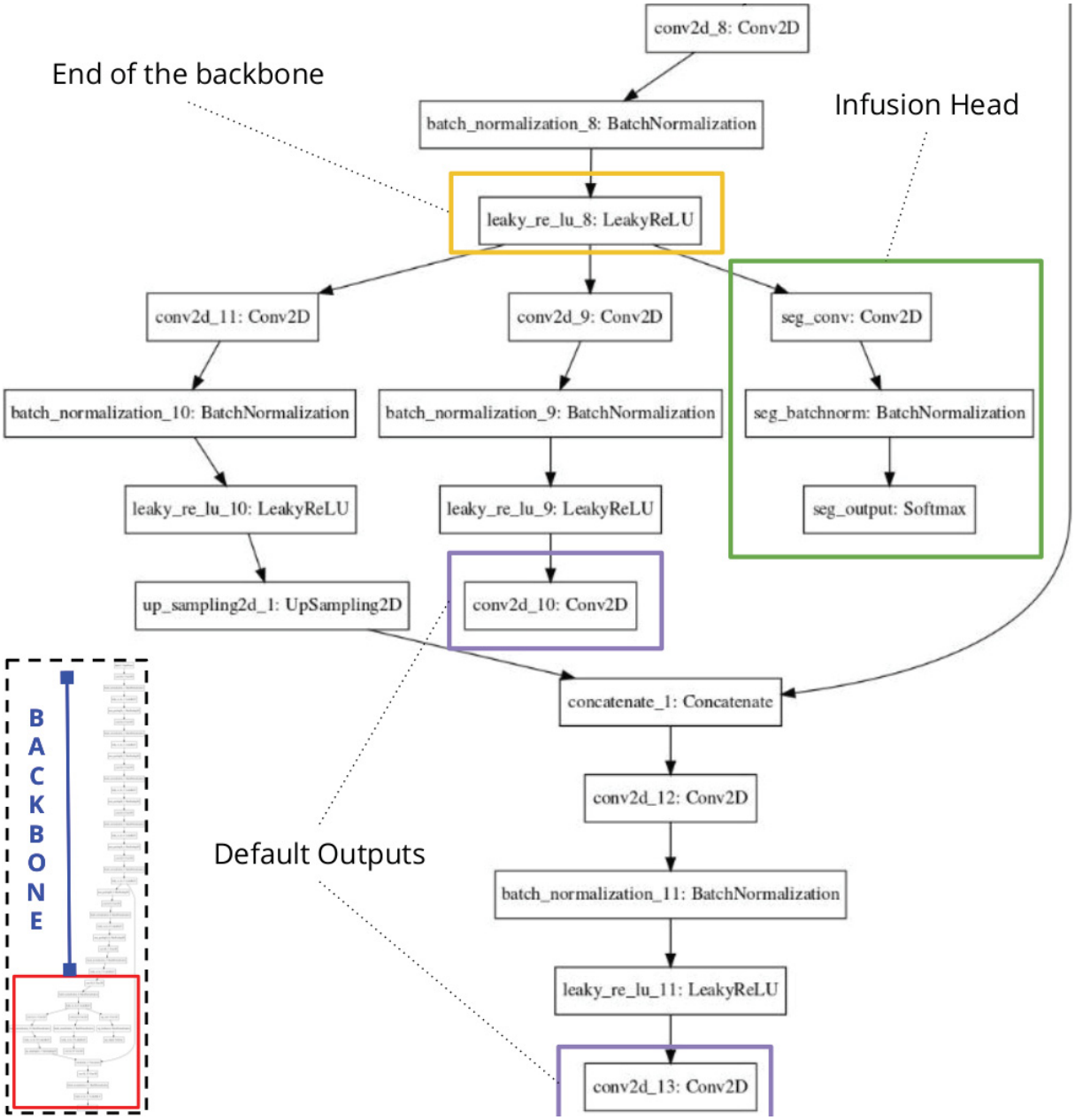


Figure 5.8: YOLOv3-tiny architecture with the additional semantic segmentation infusion head. The diagram displays the lower part of the network (red rectangle) which is zoomed out from the figure. The backbone is displayed only in the miniature (in blue). The end of the backbone is highlighted in the yellow rectangle. Three heads are connected to the backbone: two are the default for predicting bounding boxes (purple), and the last is the semantic segmentation for the infusion method (green).

Because there is a new head in the network, the loss calculation is required to be updated. Instead of changing the YOLOv3's loss function, the loss of the new head is calculated using the binary cross-entropy and is summed to the YOLO's loss value. The binary cross-entropy is a well-known loss function which calculates the performance of a model when it generates outputs which correspond to probabilities presented in values from 0 to 1. The loss value generated

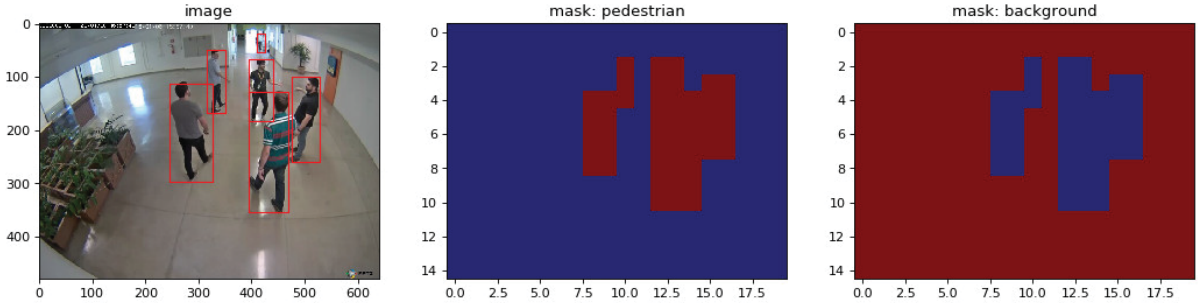
by the function increases according to the divergence to the grounding truth values, that is, the desired and correct outputs. The binary cross-entropy can be calculated using the Equation 5.1.

$$-(y \log(p) + (1 - y) \log(1 - p)) \tag{5.1}$$

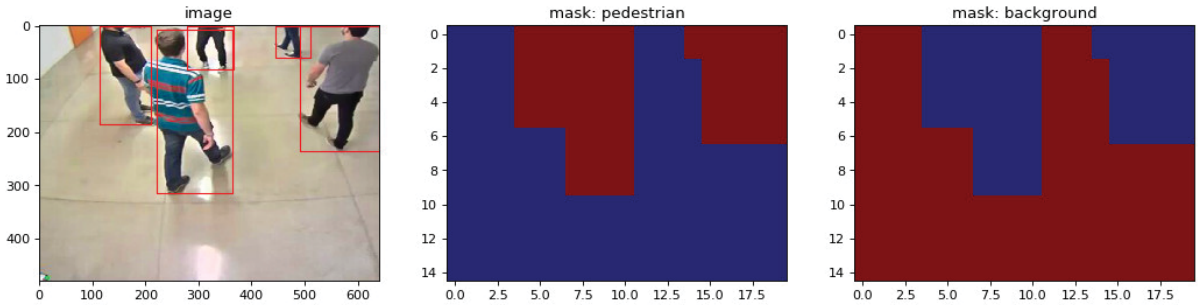
where the number of classes must be two, y is the binary assertiveness indicator (zero for incorrectly classified or one for correctly classified), and p indicates the probability of the prediction.

A weighting policy usually supports the operation of summing loss values of different prediction heads. That is, each output head may produce its loss with the possibility of defining different importance for each of them. Brazil et al. (2017) give to the semantic segmentation head a weight of 0.2 and a weight of 1.0 for the default bounding box prediction. In other words, the SDS-RCNN trains reducing the impact of the infusion in the learning process to $\frac{1}{5}$ which may have been found by the authors as the ideal fraction to give enough clue without compromising the bounding box predictions. This regulates the influence of each head while updating the network in the learning process. In our proposal, the same weighting method is applied to control the importance of the new inserted head in the YOLOv3 network. Due to the differences in the architecture, for this experiment, the weighting fraction may be different compared to the SDS-RCNN.

The final step to conclude the infusion insertion in the YOLOv3 is by providing an automatic semantic segmentation masks generation for grounding truth. In the same way, Brazil et al. (2017) explained that most of the pedestrian datasets do not contain semantic segmentation masks, adn we suffer from the same situation. We are also utilizing the Caltech Pedestrian Dataset, and additionally the PTI01, wherein both do not provide such masks. To circumvent that, the SDS-RCNN authors proposed the utilization of "weak" semantic segmentation masks, that is, masks with much fewer details about the pedestrian shape. These masks have shown to be effective because the infusion head is connected to a lower part of the backbone where the original image dimensions have been considerably reduced. With that, the masks would need only to have enough resolution of the pedestrian shape according to the already low-resolution outputs from the infusion head. This effect has been already presented in Section 3.2.5 by Figure 3.6. The authors demonstrated that creating the masks with the exact shape of pedestrian bounding boxes are very similar to the highly detailed masks at the layer in which the infusion head creates the predictions. In our implementation, we create a mechanism that automatically feeds the infusion head with ground truth semantic segmentation masks according to the bounding box annotations of the pedestrians. Figure 5.9 presents examples of the masks generation. Due to the fact the infusion head generates the output in a down-sampling factor of 32 in relation to the input image, the mask is created with a lower resolution.



(a) Left: original image with bounding box annotations with original dimensions. Middle: mask where the hot color represents the pedestrians in the dimensions to match the infusion head output. Right: the inverse of the first mask. Note that due to the lower resolution in the masks, the small-scale pedestrian in the top has been lost in the representation.



(b) Same example as Figure “a” but the image has been data augmented.

Figure 5.9: The figures demonstrates the semantic segmentation masks automatic generation to be used as grounding truth in the infusion experiments. Figure “b” demonstrates that the method supports data augmentation. The masks are generated in the dimensions compatible the the layer in which the infusion head elaborate the predictions. In this examples the input has 640×480 pixels which translates to a 20×15 mask, once the down-sampling factor of the YOLOv3 network in the place the head has been connected is 32.

6 EXPERIMENTS

This chapter presents the general configuration and details of the experimentation structure. Section 6.1 explains the hardware and software environments. The results of this work are reported by metrics defined in Section 6.2. Section 6.3 explains which works our experiments are going to be compared for each dataset. Both PTI01 and Caltech pedestrian datasets are reviewed in Section 6.4. The general training parameterization is described in Section 6.5. Lastly, Section 6.6 presents the statistical test utilized in the reporting of some results.

6.1 EXPERIMENTATION ENVIRONMENT

The YOLOv3 is built originally over the Darknet[1] open source framework. It is written in languages C and CUDA. In the early stages of this work, we have found difficulties in manipulating some pieces of the framework's code, as well as the integration with external scripts. Despite the existence of some wrappers written in the language Python for the Darknet, we decided to use another implementation of the YOLOv3, entirely written in Python. We have chosen to use the *keras-yolo3*[2] built over the Keras[3] with TensorFlow[4] backend. This choice enabled us to quickly prototype and test many experiments, avoiding the need to manipulate C code and also due to the high-level Application Programming Interface (API) for neural network provided by Keras. Many modifications have been done to the implementation and have been re-published[5].

Despite the fact that we have been using a different framework than the original, it is possible to share the model between the Keras and Darknet through parser scripts. That is how we have been able to reuse the pre-trained weights from YOLOv3. This is very important once it is known that Python is slower than C code. Thus, for evaluation of detection speed, we can run the model over Darknet and reach the highest processing performance.

The experiments have been executed on a server with two AMD Ryzen 1920X 12-Core 3.5GHz Processors, 64GB RAM, an NVIDIA TITAN X 12GB GPU, and an NVIDIA TITAN XP 12GB GPU. We gratefully acknowledge the support provided from NVIDIA Corporation through the donation of the TITAN X GPU. At the software level, there was Ubuntu 16.04 x64, CUDA 9.0, Python 3.5.2, Keras 2.2.2 and TensorFlow (GPU version) 1.10.0.

6.2 METRICS

In order to evaluate the results of each experiment with the model, it is necessary to utilize some metrics. One of the most basic metrics related to Object Detection is the *Intersection-over-Union* (IoU), also known as the *Jaccard Index*. This metric measures how spatially similar a predicted bounding box is to a given Ground-Truth (Figure 6.1) (Rahman and Wang, 2016). Equation 6.1 demonstrates the calculation for the IoU, which defines the proportion between the intersection of both bounding boxes over their union. If both bounding boxes (prediction and ground-truth) have the same dimensions and are in the same location in the image, the resulting

[1] https://pjreddie.com/darknet
[2] https://github.com/qqwweee/keras-yolo3
[3] https://keras.io
[4] https://www.tensorflow.org
[5] https://github.com/gustavovaliati/keras-yolo3

value is going to be 1.0 (100%, or perfect prediction), and if there is not a single pixel intersecting the value is going to be zero (0%).

$$IoU = \frac{\text{pixels in the region of intersection}}{\text{pixels in the union of both regions}} \tag{6.1}$$

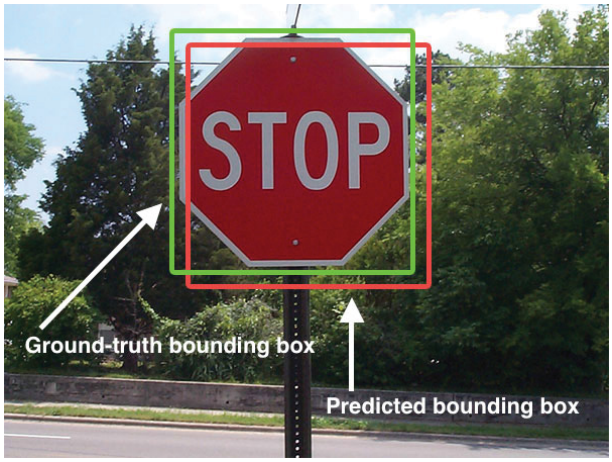


Figure 6.1: Example of a predicted bounding box and ground-truth bounding box. Source: www.pyimagesearch.com by Adrian Rosebrock.

There are common terms in the training and testing of neural network models which refers to the learning samples and the predictions. The Ground-Truth (GT) (a.k.a. Real Positive (RP)), in Object Detection, corresponds to the bounding box in which the model will be training. They represent the perfect results that the model must pursuit while training. The metrics use these GT "annotations" in the dataset as the referential. The True Positives (TPs) refers to predictions that have been correctly done and are bond to GTs. A False Positive (FP) is a wrong prediction. In Object Detection, an FP is usually related to determining the existence of an object in an invalid location in the image, or correctly finding the location but wrongly predicting its class. The False Negative (FN) corresponds to every GT that the model could not localize.

The TPs are typically determined with the help of an IoU threshold. The algorithms that evaluate the predictions will define each of them as TP or FP according to the minimum IoU established in the metric. For instance, if the IoU threshold is set to 0.5 (50%), the predicted bounding boxes will be marked as True Positive only if they have a IoU greater than or equals to the threshold (0.5). Otherwise, it will be defined as False Positive. In other words, this threshold defines the minimum spatial accuracy wanted to define a prediction as correct. Another important criterion to define a prediction as True Positive in object detection challenges like Caltech Pedestrian Benchmark, is that only one prediction can be attached to a Ground-truth. That means even if three predictions had enough IoU for a single Ground-truth, two of them would be marked as False Positives due to redundancy.

According to Powers (2011), *Recall* and *Precision* are metrics commonly utilized to evaluate the results of Machine Learning experiments. The Recall, defined by Equation 6.2, demonstrates the relationship between the True Positives (the positives predicted as positives) over the Real Positives (the ground-truth of positives), which measures the ability in finding all the positive cases.

$$Recall = \frac{\text{True Positives}}{\text{Real Positives}} \tag{6.2}$$

The *Precision*, defined by Equation 6.3, presents the proportion between the True Positives over the sum of the True Positives and the False Positives, that is, measures how many cases are truly positives between the ones predicted as positives.

$$Precision = \frac{\text{True Positives}}{\text{True Positives + False Positives}} \tag{6.3}$$

Another important metric is the Average Precision (AP), used in object detection challenges such as COCO and PASCAL VOC[6]. According to Yilmaz and Aslam (2006), AP is a system-oriented, stable and highly informative measure for retrieval effectiveness, being one of the most frequently used and referenced metric. The AP is calculated for each class individually in the dataset, but it is common to present the mean over every AP from each class, the "mAP." A Precision/Recall curve is computed where the outputs are ranked according to the prediction confidence. The AP becomes defined by the mean precision at a set of eleven equally spaced recall levels from zero to one, which defines the shape of the Recall/Precision curve (Equation 6.4). The Precision (p) is calculated in each level of Recall (r) by interpolation, that is the maximum Precision corresponding to Recall exceeds (r) (Equation 6.5) (Everingham et al., 2010).

In the COCO benchmark (Lin et al., 2014) there are multiple "mAP" variations, such as mAP-50 and mAP-70. The number in the metric name (50 or 70) is related to the previously explained minimum acceptable IoU threshold. This means the mAP-50 metric, which is the conventional one, is a bit more tolerant in terms of detection quality than the mAP-70. The mAP-50 requires that the predicted bounding boxes have at least 50% in the IoU score according to the ground truth annotations, while the mAP-70 is more rigid requiring 70%.

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} p_{interp}(r) \tag{6.4}$$

where AP is the Average Precision; r is the Recall; $p_{interp}(r)$ is the interpolated precision at each Recall level.

$$p_{interp}(r) = \max_{\tilde{r}:\tilde{r} \geq r} p(\tilde{r}) \tag{6.5}$$

where $p_{interp}(r)$ is the interpolated precision at each Recall level; r is the Recall; and p is the Precision.

In some experiments in which we used multiple classes, the mAP has heavily degraded by low scores of a few classes. Because that is regular arithmetic mean such behavior is expected. However, the classes with very low AP, had just a few samples, while the classes with high score had thousands (example in Figure 6.2). We decided to calculate a weighted mean instead of an ordinary one, according to the number of samples for each class. We named such metric as weighted mean Average Precision (wAP), and it did give a better representation of the final score. Equation 6.6 explains the wAP calculation. Additionally, we have introduced the General Recall (GR) as an extra metric that gives a better overview of the model's Recall ability independently of individual class performances. The GR is the ratio of the total number of True Positives of all classes by the total number of ground truth objects.

$$\text{wAP} = \frac{\sum_{i=1}^{n} g_i \text{AP}_i}{\sum_{i=1}^{n} g_i} \tag{6.6}$$

[6]http://host.robots.ox.ac.uk/pascal/VOC/

where n is the number of classes; g is the number of ground truth objects for the respective class; and the AP is the Average Precision of that class.

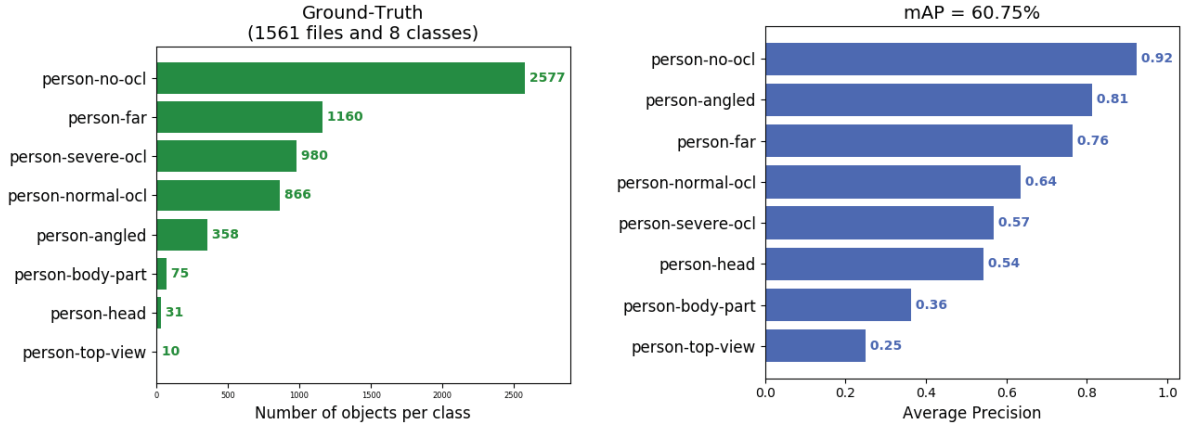


Figure 6.2: The charts exemplify the motivation of the wAP. For instance, it is possible to see (on the right) the class "person-top-view" with AP in 0.25 is the one with the lowest number of GT (10 instances displayed in the left), while the best-scored class has more than 2500 instances.

In the Caltech Pedestrian Detection Benchmark, instead of using mAP, the authors define a different metric: the LAMR accompanied by the False Positives per-image (FPPI) × Miss Rate curve (Dollár et al., 2009b). According to Dollár et al. (2012), some tasks usually accept an upper limit of FPPI rate independently of the pedestrian density, such as tasks in automotive applications. Because of that, they prefer this metric instead of the Precision/Recall curves. The metric uses the LAMR which is conceptually similar to the AP from the PASCAL VOC challenge. The LAMR is used to summarize the detector's performance, by calculating the average of miss rates at nine different FPPI rates, equally spaced in log-space of range 10^{-2} to 10^{0} (Dollár et al., 2012).

In this work, we have modified[7] an independent implementation[8] of the PASCAL VOC mAP for automation purposes and additional charts generation, which also provides the LAMR metric. On Caltech, to calculate the LAMR and FPPI × Miss Rate curve, we use the default evaluation toolbox provided by the Caltech Benchmark which runs on MATLAB.

6.3 BASELINES

It is common to have baselines for the experiments, that is, solid results used in the same context as a reference for general comparison. The results reported in our work are generated through experimentation in two different datasets: Caltech Pedestrian Dataset and PTI01 Pedestrian Dataset. In this section, we briefly report the main baseline results.

6.3.1 PTI01 Pedestrian Dataset

In the case of PTI01 dataset, the existing baselines are related to our paper (Valiati and Menotti, 2018) which reports the performance over YOLOv2 (Redmon and Farhadi, 2017), SSD (Liu et al., 2016) and Faster R-CNN (Ren et al., 2015). The three models have been tested with compatible pre-trained weights on the first version of the PTI01. The primary objective was to evaluate the discrepancy between the models regarding speed and the generalization power from

[7]https://github.com/gustavovaliati/mAP
[8]https://github.com/cartucho/mAP

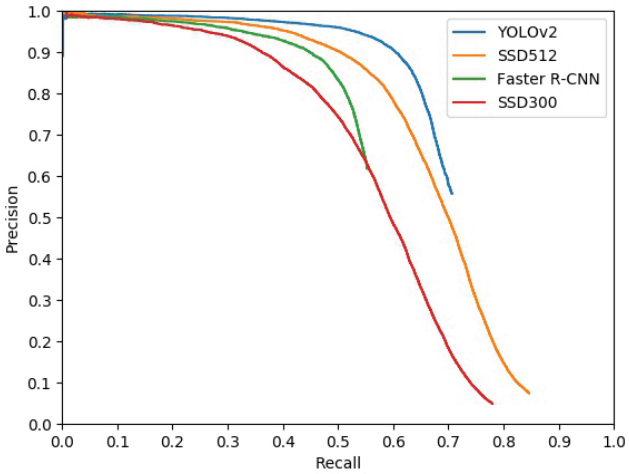


Figure 6.3: (Valiati and Menotti, 2018) Precision × Recall for each model. YOLOv2 holds the best Precision while increasing Recall.

the originally trained scenarios. Table 6.1 presents the achieved results in the mAP and also the FPS to verify the speed performance. According to the results, we see that YOLOv2 did achieve the best accuracies and speeds for both datasets, unless for speed in SSD300 (Liu et al., 2016) with PASCAL dataset. In Figure 6.3 the Precision × Recall curve is plotted. It demonstrates that YOLOv2 (Redmon and Farhadi, 2017) can hold a higher Precision while increasing the Recall.

In summary, the paper demonstrates that YOLOv2 can potentially transfer better the learned knowledge from one dataset to the other, without any fine-tuning procedures. Despite being a valid experiment, for this work, such results are not so relevant anymore for three reasons: 1) The PTI01 dataset has evolved to its version 3, with a higher quality of annotations, while the paper has experimented over the first version; 2) YOLOv2 (Redmon and Farhadi, 2017) has also evolved, and in this work, we approach its newest version: the YOLOv3 (Redmon and Farhadi, 2018); 3) The achieved scores by Valiati and Menotti (2018) have been generated to evaluate a different aspect of the models, where no fine tuning have been applied. In this work, such an evaluation is not part of the main goal.

Table 6.1: (Valiati and Menotti, 2018) Evaluation results: PTI01 and PASCAL VOC. YOLOv2 achieved the best accuracy in both datasets, while losing in speed just for SSD300 in PASCAL VOC.

Model	**Dataset**	**Metrics**	
		mAP	**FPS**
Faster R-CNN	VOC	73.2	7.0
	PTI01	51.6	3.6
SSD 300 × 300	VOC	74.3	**46**
	PTI01	55.9	4.9
SSD 512 × 512	VOC	76.8	19
	PTI01	65.6	4.1
YOLOv2 544 × 544	VOC	**78.6**	40
	PTI01	**67.1**	**10.9**

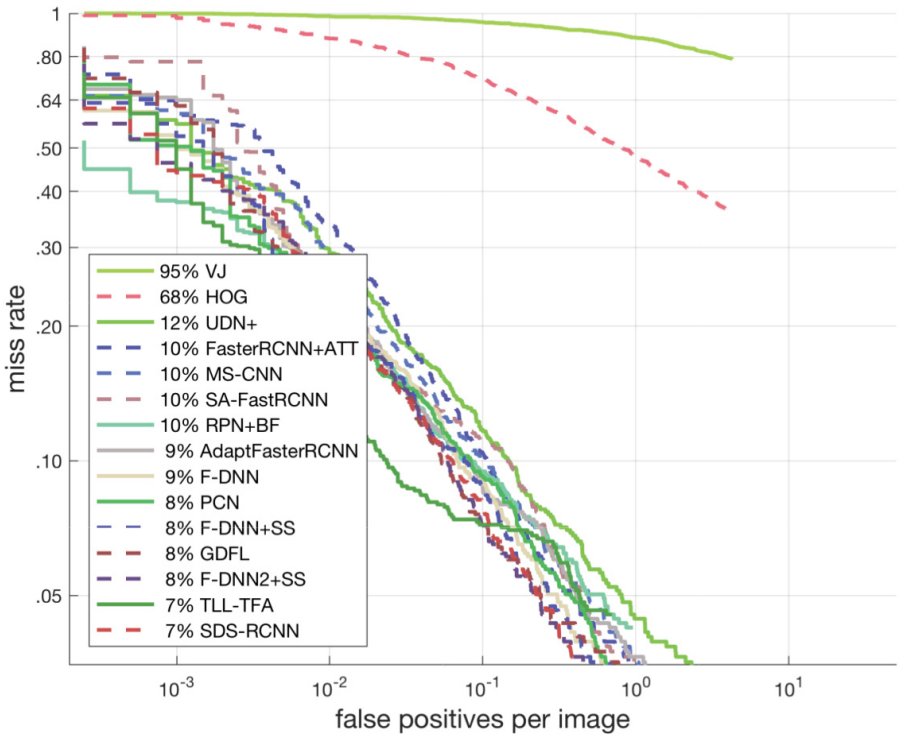


Figure 6.4: Caltech Reasonable Results. Source: http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians

6.3.2 Caltech Pedestrian Dataset

Considering the Caltech Pedestrian Dataset we have chosen, in our interpretation, two works that are in the state-of-the-art of the Caltech's challenge (Dollár et al., 2012). Having this baseline, it gives us a referential to analyze the performance achieved by our experiments in this dataset.

According to the "Reasonable" setting defined by the authors of the challenge, the SDS-RCNN (Brazil et al., 2017) and TLL-TFA (Song et al., 2018) lead the accuracy rank (Figure 6.4). Both works are technically explained in Chapter 3. This setting evaluates the performance of Pedestrians equals or taller than 50 pixels, and with partial occlusion or no occlusion at all. In other words, this is an evaluation of feasible difficult scenario, excluding Pedestrians that are too small or too obstructed in the image by other objects.

Beyond the Reasonable setting, the challenge defines at least nine other evaluation scenarios (settings), which are briefly explained in Section 6.4. Among the many works reported in each of these scenarios, we verify that TLL-TFA leads in 6 of them and is well ranked in the other 3, probably being the best method in the challenge. SDS-RCNN is competitive in those other settings, leading in the one which TLL-TFA does not go so well (Partial Occlusion) and having top-tier results in other 3.

The choice of these two works covers all the settings evaluated by the Caltech Benchmark. When achieving results close to or greater than this baseline, they will define the experimented model as competitive to the state-of-the-art.

The authors of this work find relevant, but not essential, to evaluate SDS-RCNN (Brazil et al., 2017) and TLL-TFA (Song et al., 2018) on the PTI01 dataset, creating an additional baseline. However, for this work, this task demonstrated not possible for two reasons: 1) TLL-TFA does make the source code publicly available. It would be necessary to request such access, analyze the framework and them try to reproduce in the local dataset; 2) SDS-RCNN, despite providing the source code, the required software depends on licensing (MATLAB) and a meticulous technical

process trying to reproduce it in our environment. In that way, we choose to keep both works only in the Caltech scenario with their official results.

6.4 DATASETS

For the general experiments, two datasets have been used: PTI01 Pedestrian Dataset and Caltech Pedestrian Dataset. The PTI01 is introduced by the authors and is used as the primary dataset for experimentation, while in Caltech we do experiment the most stable models. A brief review of both datasets is presented in this section.

6.4.1 Caltech Pedestrian Dataset

Dollár et al. (2009b) introduced the Caltech Pedestrian Dataset with the idea of providing a more challenging scenario compared to the most popular ones at the time, such INRIA (Dalal and Triggs, 2005). The authors wanted to increase the variety of pedestrian scales, poses, and occlusions, as well making it bigger than others. The authors define that the variation in appearance and the vast amount of images for training as important for evaluating Pedestrian Detectors. With the dataset, they release a new evaluation metric called FPPI × Miss Rate (explained in Section 6.2). Figure 6.5 illustrates some samples from the dataset.

Figure 6.5: Caltech samples. The dataset is composed of video recorded from a moving vehicle in an urban scenario. The green bounding boxes are the annotation with the full extent of the pedestrian, and the yellow ones represent the visible part. Source: (Dollár et al., 2009b)

The Caltech dataset was built by the acquisition of ~10 hours of video in the resolution of 640×480 pixels from a moving vehicle in an urban environment. The total number of frames collected are of ~1 million, with 250 thousand being annotated. The annotations are composed by 350,000 labeled pedestrians, which represent 2,300 unique people.

The annotation was made in the format of a bounding box, that is, a rectangle that delimits the extremities of the pedestrian body. However, each pedestrian receives two annotations. One annotation refers to the visible part of the pedestrian. The second contains the visible part and an estimation of the entire pedestrian when occluded by some other object (Figure 6.6). This double annotation for a single pedestrian enables the calculation of occlusion through the analysis of the proportion visible and the estimated one.

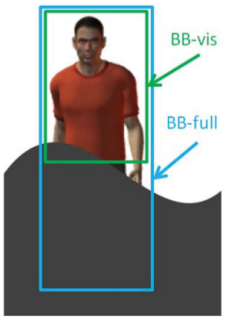


Figure 6.6: Caltech bounding box types. The green bounding box (BB-vis) represents the visible part of the pedestrian, while the blue (BB-full) represents the estimated full body. Source: (Dollár et al., 2012)

Each pedestrian has been annotated in one of the following classes: "Person" for individual pedestrians; "People" for a group of pedestrians which the annotators did find too laborious to annotate unitarily; or "Person?" if the annotator did not have a clear identification of the pedestrian. The majority of annotations (~80%) are in the "Person" class.

The Caltech authors (Dollár et al., 2009b) have created a labeling tool in order to help the annotators in the task of labeling such big dataset. The tool required from the annotators sparse labeling in a video scene, that is, annotating just a few frames evenly-spaced with non-annotated ones. The tool automatically predicted and annotated the intermediate frames. Even with this tool, it was necessary approximately 400 hours of work to finish the labeling.

Dollár et al. (2012) elaborated a study of many characteristics of the Caltech dataset, such as patterns in the occlusions, scales and positions of the pedestrians. As an example, we show in Figure 6.7 four analysis of statistics about the scale.

In our work, we choose two different training and testing compositions for evaluation. The Caltech dataset contains 11 sections of data where the first 6 (known as "S0-S5") are normally used for training, and the remaining (five sections known as "S6-S10") are, by default, separated for testing. The dataset has 4 official evaluation scenarios: 1) "ext0": training over any external data and testing the model with "S0-S5"; 2) "ext1": same as "ext0" but testing on "S6-S10"; 3) "cal0": 6-fold cross validation over "S0-S5"; 4) "cal1": train on "S0-S5" and test using "S6-S10".

Instead of following that strict scenario configuration, we select the frames in the same way Brazil et al. (2017) did. They have used a configuration named by Zhang et al. (2016b) as "Caltech10×". It is a sampling from the training set ("S0-S5"), considering only every 3rd frame totalizing 42,782 images. That is an increase of 10× compared to the conventional method which considers every 30th frame, called "Caltech1×". For the testing phase, they use 4,024 images according to the "Caltech1×" setting.

Dollár et al. (2009b) have elaborated an evaluation methodology which reports the results in 10 different settings. These settings propose the evaluation of different characteristics within the pedestrians in the dataset. The summary of such settings is presented in Table 6.2. As Brazil et al. (2017), we focus on the "Reasonable" settings for the evaluation.

For further experiments, we choose an additional configuration for the training set proposed by Zhang et al. (2016b). They have created an improved version of the annotations for training and testing using the "Caltech1×" sampling. The improvement proposed by them is related to some errors in the original annotations mostly caused by the automatic labeling tool. The predictions produced by the tool, through the use of sparse annotations, could not handle all the situations where the pedestrian is not present anymore in the image due to some change in the environment. For instance, a car passing in front of a pedestrian like in the first image of

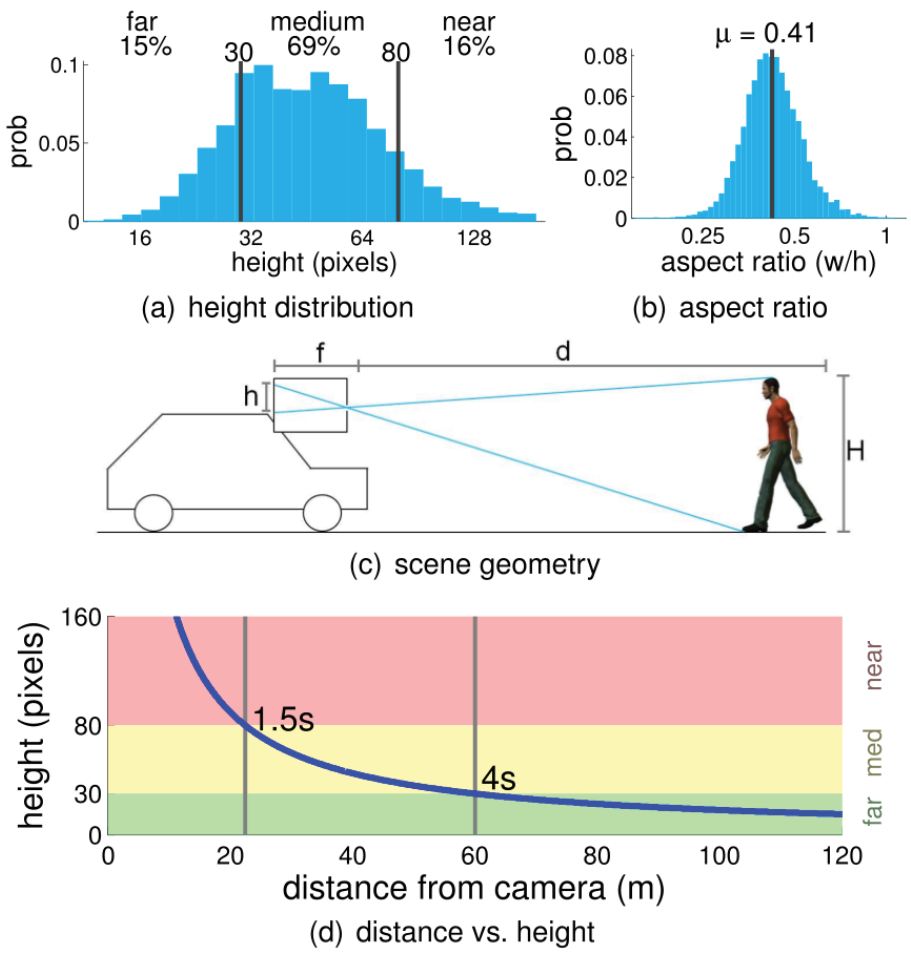


Figure 6.7: Caltech statistics example. The images (a) and (b) demonstrates the height and aspect ratio distributions respectively. The image (c) shows that, in the dataset scenario, the pedestrian height (in pixels) is inversely proportional to the distance of the camera. In the image (d) they establish the relationship between the pedestrian height (in pixel) with how much time it takes to the car (in 55km/h) reaching him. Source: (Dollár et al., 2012)

Table 6.2: Caltech Default Evaluation Settings.

Setting	Description
Reasonable	Occlusion: none or partial; Height >= 50 pixels;
Overall	Entire dataset;
Typical aspect ratios	Occlusion: none; Height >= 50 pixels; Std. Dev. of mean aspect ratio <= 0.1
Atypical aspect ratios	Occlusion: none; Height >= 50 pixels; Std. Dev. of mean aspect ratio >0.1
Near scale	Occlusion: none; Height >= 80 pixels;
Medium scale	Occlusion: none; Height >30 and <80 pixels;
Far scale	Occlusion: none; Height <= 30 pixels;
No occlusion	Occlusion: none (0%)
Partial occlusion	Occlusion: partial (1-35%)
Heavy occlusion	Occlusion: heavy (35-80%)

Figure 6.8(a)). The tool also could not handle the natural motion of “up and down” from walking pedestrians, generating a slight offset in the bounding boxes positions (Figure 6.8(b)).

(a) False annotations (b) Poor alignment

Figure 6.8: Problems in Caltech annotations. The image (a) contains annotations with no pedestrians which have been removed by Zhang et al. (2016b). In the image (b), in red, are represented the dislocated annotations due to pedestrian motion and in green the corrected version. Source: (Zhang et al., 2016b)

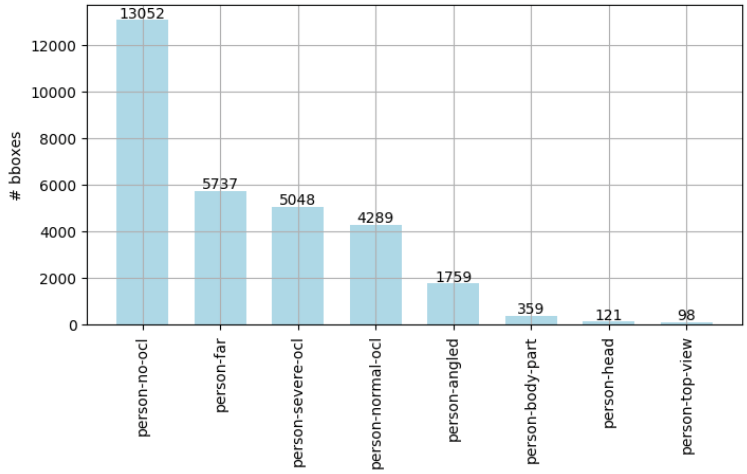


Figure 6.9: PTI01 class distribution.

6.4.2 PTI01 Pedestrian Dataset

The PTI01 Pedestrian Dataset, proposed in Chapter 5, is built with the main interest of evaluating pedestrian detectors in a real-world scenario well-known by the authors of the current work: the video surveillance network from the Itaipu Technological Park (PTI). The dataset is composed by 7,927 images collected from 21 strategical cameras which are part of a surveillance network with more than 250 cameras. The frames from each camera belong to events, wherein these events correspond to continuous sequences of movement on the scene. There are 30,463 pedestrians annotated, spread in 8 different classes (Figure 6.9). The scenes are recorded in ~2.5 FPS.

The dataset has three versions. The first one, evaluated by Valiati and Menotti (2018), has a single class and had been part of the labeling done by the authors of the current work and the remaining by the online platform called "Hive"[9]. In the second version, a few volunteers collaborated with the annotations mostly replacing the labels given by the online platform, increasing the annotations quality by the end. The last version introduced the eight classes as part of our objective of analyzing the behavior of the pedestrian detector by the isolation of bounding boxes with similar patterns (intra-class variations).

In terms of training and testing sets, the dataset has been configured in 4 different ways. These configurations have been proposed to provide an analysis of possible different levels of difficulty. The configurations are:

[9]https://thehive.ai/

- "High-bias": 20% of the frames from each event of each camera is separated for testing only. The remaining goes to the training set. We call it "high-bias" because frames from the testing set have a high probability to be very similar to the trained ones. There is a chance of training over four frames and testing in one that is very close or between them. Because of that, we expect the model to learn easily;

- "Leaving cameras out": contrary to the "high-bias" configuration, this one is projected to be harder. We intend to check how hard it is to the pedestrian detector to generalize the learned knowledge to an unknown camera scenario. We take images from each of the 21 cameras and calculate the discrepancy in appearance between them. Each scene (camera) is compared to each other using two metrics: "Mean Squared Error"(MSE) which is the sum of the differences between two images; and the "Structural Similarity Index"(SSI) that additionally takes textures in the account to calculate differences. Then, two ranks are created by the sum of the differences of each metrics and by the end it is possible to define the most similar and most discrepant camera scenarios. In this configuration, we create three different sub-configurations separating all the frames of chosen cameras for the testing set. Those sub-configurations are as follow: 1) the most discrepant camera is chosen for test and the remaining of the frames (all other cameras) for training; 2) the three most different cameras are separated as well; 3) the five most discrepant are separated.

- "Leaving events out": After preparing two configurations in which one should be easier and the other harder, we try to project a new one for moderated difficulty. All the events in the dataset are sorted chronologically. For any camera that has a single event, the camera is separated for training. For those that remain (majority), we choose the first event of that camera and place them in the testing set. The remaining goes for the training set. In that way, we expect to have training data from all the cameras, but at the same time we raise the difficulty by testing with frames from events that the model have never seem;

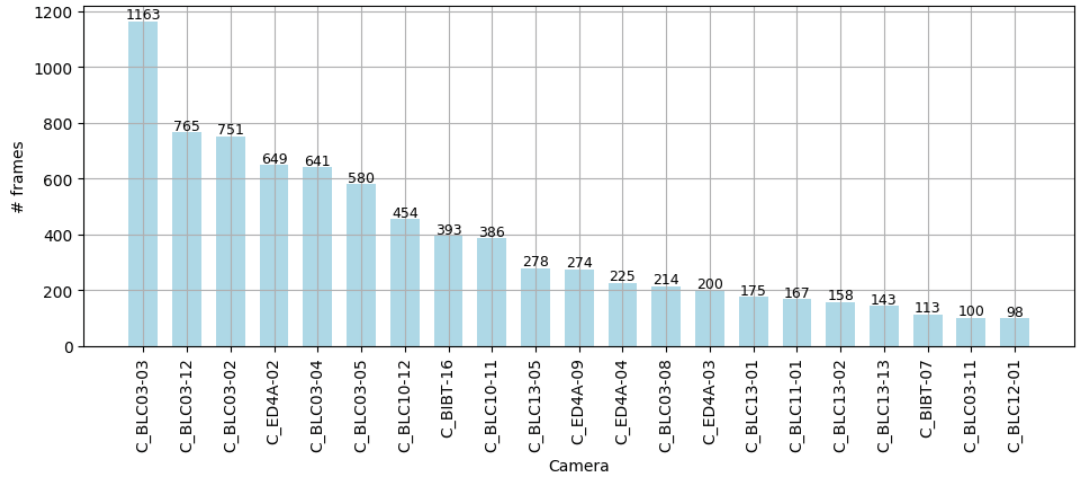
- "Leaving cameras out & discarding frames": this is a configuration planned to be the hardest among the others. The idea is the leave some cameras out, by default 3, and at the same time, we discard 66% ($\frac{2}{3}$) of the training dataset (keeping only every 3rd frame). This would provide to the detector very scarce data, and we would evaluate its abilities in learning in such a difficult situation;

Table 6.3 summarizes the training/testing distribution for the different configurations. More information about the dataset is described in Chapter 5. Figure 6.10 presents the distribution of frames and bounding boxes quantities for each camera. The general aspect ratio and the specific values for each class are presented in Figure 6.11. The changes in appearance for the created classes do reflect changes in the pedestrian shape through the analysis of their aspect ratio. Such variations could be taken in consideration while experimenting the dataset. Figure 6.12 presents the number of frames that contains zero to ten (maximum) people annotated per scene. It is possible to see that the majority of frames (2219) contains five people on the screen, only 29 images with ten people and 269 frames with no people at all.

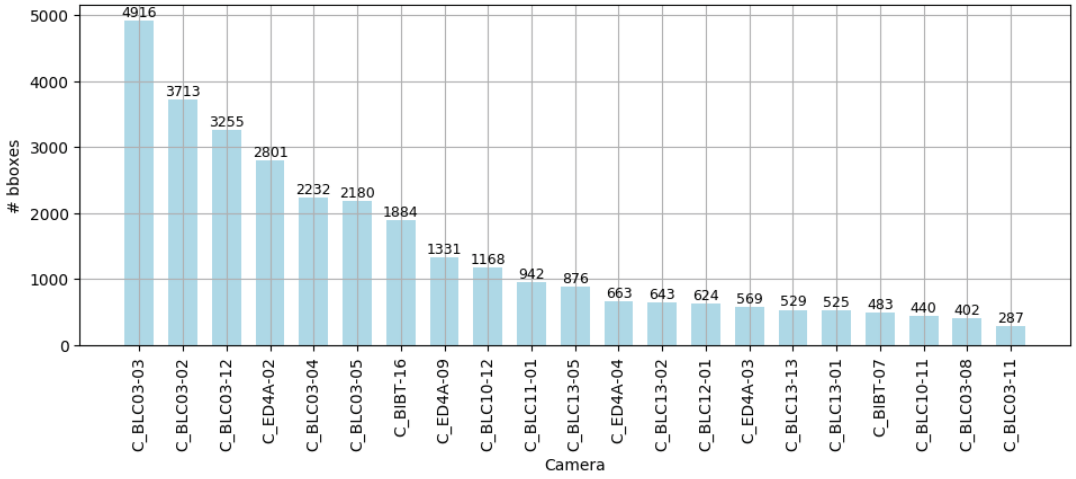
Figure 6.13 shows the distribution of pedestrian locations from the PTI01 dataset across the image dimensions (640×480 pixels) using a "heat map." The heat map points in hot colors the locations in the image which the bounding boxes more occurs, while the colder colors correspond to fewer occurrences. The "overall" heat map demonstrates the main locations of all pedestrians which tends to be mostly centered and slightly upwards. The corners and sides present lower

Table 6.3: PTI01 train/test configurations

Configuration	Frames	
	train	test
high-bias	6366	1561
leaving 1 camera out	7727	200
leaving 3 cameras out	6864	1063
leaving 5 cameras out	6141	1786
leaving events out	6521	1406
leaving 3 cameras out & discarding	2288	1063



(a) Frames by camera



(b) Bounding boxes by camera

Figure 6.10: PTI01 frames and bounding boxes by camera

concentration of pedestrians across the entire dataset. The figure also shows the heat map by class. The "angled class" are located in the left-center and right-center in the image. The "body-part" are majority correspondent to pedestrians in the extreme edges of the scene, while the "far" concentrates at the top, the "person-head" and "top-view" the bottom. The "no-ocl" (no

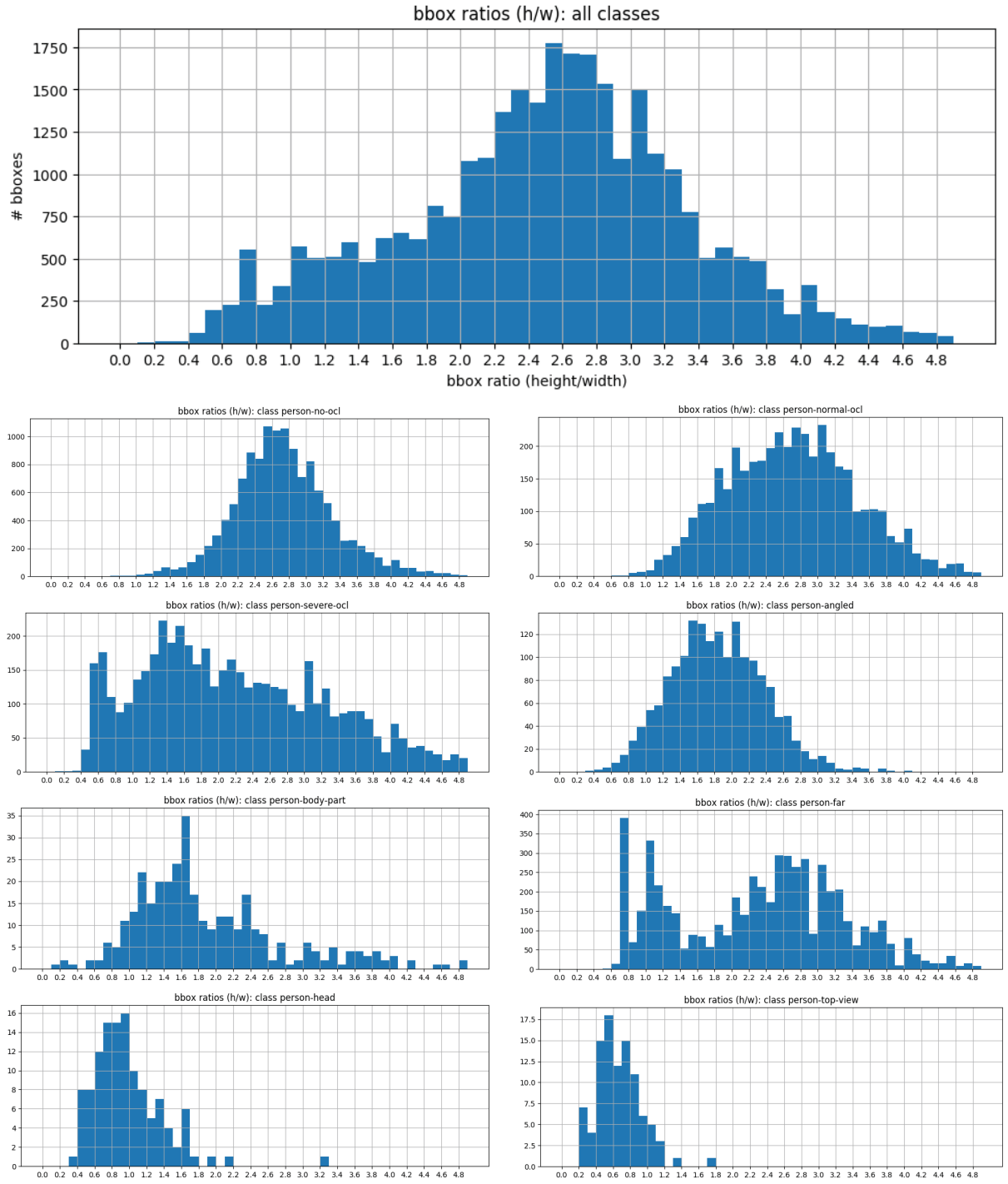


Figure 6.11: PTI01 bounding boxes ratios. It is possible to verify a variation in the aspect ratio distribution over different classes which demonstrate that the bounding boxes vary in shape according to the pedestrian view or position.

occlusions) contains pedestrians very centered in the image, while the "normal-ocl" starts to spread the concentration and the "severe-ocl" demonstrates a general distribution with a slight concentration on the image left and right edges. The rectangle areas in the heat map which are somehow highlighted are normally referent to pedestrians that stay in the same position in the scene for a long time. The red rectangle in the "severe-ocl" class heat map corresponds to a "receptionist" that stay still in the scene in every frame. The figure also presents heat maps for 10 of the 21 cameras from the PTI01 dataset. With each heat map, the correspondent scene (on the

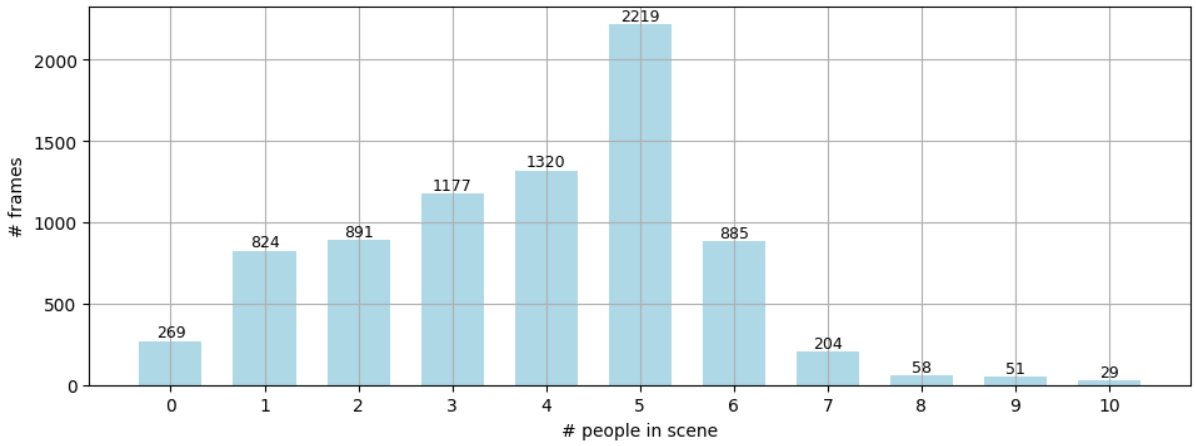


Figure 6.12: PTI01 distribution of frames by the count of people in scene. The majority of the frames (2219) contain 5 people in scene.

left) is shown to enable a better understanding of the heat pattern displayed. For instance, the camera "C_BLC13-13" demonstrates that the pedestrians are highly concentrated on the top right of the image due to distance and a narrower path, and the heat pattern informs a flow that goes to the bottom-left of the image with a more disperse distribution due to the closer corridor view.

6.5 TRAINING

The YOLOv3 training has been executed using a public implementation[10] based in the Keras and TensorFlow frameworks. By differing from the original framework (Darknet), many training parameters have been updated to fit the framework logic. Despite our interest in fine-tuning those parameters to fit the current training context better, we have found that, mostly, the default parameterization was enough. In this section, we briefly report some important parameters as well the reason we have chosen PTI01 to support the majority of the experiments. We denote that not all the detailed results are presented for the sake of brevity.

The default "initial learning rate" defined in the YOLOv3 Keras implementation is 10^{-4} and demonstrated to be the most efficient on PTI01 according to our results reported in Table 6.4. Additionally, we apply the "learning rate reduction" multiplying the current learning rate value by a factor of 0.1 if the model runs three epochs without reducing the "validation loss."

Table 6.4: Initial Learning Rate comparison. The default value of 10^{-4} reports better results on PTI01.

Initial Learning Rate	**mAP**	**epochs**
10^{-3}	0.6791	54
10^{-4}	**0.6957**	46
10^{-5}	0.6795	39

Inspired by Masters and Luschi (2018), the models have been experimented with different batch sizes. The tested quantities were 2, 4, 8, 16 and 32. The general understanding of the achieved results (mAP) for each batch size is that higher the epoch is slower is the convergence to the best score. Using batch size as 32, it has taken many more epochs to achieve similar results

[10]https://github.com/qqwweee/keras-yolo3

performed with a batch size of 2. Because of that, we have chosen to keep the experiments using the batch size in 2.

Instead of manually deciding when to stop the training, we have made use of a default functionality available in the Keras framework that automatically interrupts the learning through pre-defined criteria. In such a way, it is not necessary to depend on a maximum epoch value, although we have defined a limit of 100 epochs, just in case. The selected parameter to stop training is based on the lack of improvement in the "validation loss," which is activated if the model runs more than five epochs without improving the loss. Our experiments demonstrated that this value was enough, and the epoch limit has never been reached.

Another relevant configuration proved and proposed by the default implementation for the training is "freezing" the network backbone's weights for a specific number of epochs. When loading the pre-trained weights in the YOLOv3's network, the last layers in each head will be randomly initialized while the remaining of the network utilizes the values from the pre-trained model. That happens because both models, ours and the pre-trained one, are in different domains and with different class configurations. Because of that, the last layers are formed by feature maps of different shapes which can not be reused. When the training starts, such random initialized weights have a deficient performance compared to the network's backbone which has inherited the good weights. This low performance in the heads of the network provokes aggressive updates in the entire network due to the calculations in the backpropagation algorithm. In the first iterations, such updates do worsen the good pre-trained weights until the heads of the network start predicting better.

Once we freeze the weights of the backbone when the training starts, and keep like that until the weights in the heads "understand" the knowledge provided by the backbone, the backpropagation algorithm will only update the non-frozen parts of the network's weights, that is, the heads. After one epoch, the heads seem to be reasonable, and we unfreeze the remaining of the network so that the updates are now more coherent. Such freezing technique enables the model in reaching a good convergence with fewer epochs because the good weights are not initially scrambled.

We report some other configurations. The Optimizer function is kept as the default "Adam Optimizer" (Kingma and Ba, 2014). The "loss function" is a re-implemented version in Keras of the original YOLOv3 function, provided by the "keras-yolo3" framework. As initial weights, we have used the one pre-trained over the COCO dataset (Lin et al., 2014), which is made publicly available by Redmon and Farhadi (2018).

6.5.1 PTI01 as the primary dataset for experimentation

The PTI01 dataset has a smaller size and provides an easier pedestrian detection task compared to the Caltech. While executing the preliminary model training, we have found impractical to experiment on Caltech for secondary modifications in the training parameters, such as learning rate, epochs and batch size.

As reported in Section 6.4, the Caltech has originally some bad quality annotations, which in our opinion could influence the preliminary experiments. To be more effective in such an issue, we also report the main results in the sanitized version of the dataset provided by Zhang et al. (2016b).

Another reason that made training on Caltech harder than with PTI01 is that due to the configuration "Caltech10×" and also due to its natural higher difficulty, the YOLOv3 (Redmon and Farhadi, 2018) takes about 2.5 days to complete the learning process. It would be impractical to retrain the model for every minor change in the configuration, being that the PTI01, depending on the training method could take from 2 to 12 hours.

Because of those motivations, we have chosen to use PTI01 as the primary dataset. Any adjustment in the model should be first stabilized in that dataset, and after a solid confirmation of the effectiveness of the modification, the experiment could be replicated to Caltech. That is the main reason we report a higher variety of evaluations for PTI01 than for Caltech.

6.6 STATISTICAL ANALYSIS

In this work, the "T-test" is utilized as a statistical test to evaluate the effectiveness of the "weak semantic segmentation infusion" technique which is applied to the YOLOv3 network in order to try to improve the pedestrian detection quality. The technique is part of the proposals of this work, and the experimentation idea is explained in Section 5.3.

The "T-test," also known as "student's T-test," is a statistical test that aims to compare the means of two groups and check if there is a significant difference between them. Willian Sealy Gosset developed it in 1908 which utilized the pseudonym of "Student" when submitting his study. It is a parametric method where the parameter of the probability distribution can be inferred and set as a variable. This test requires the data to be normalized and with equal variance. There are two types of T-tests, the "independent" and the "paired". The first can be used for groups which will be compared but are not directly related to each other, enabling inter-group comparisons. The paired T-test is related to two groups (e.g., A and B) which are dependent on each other. Usually, group B contains the same individuals initially present in group A, but they are in a post-experiment state. That is, group A has been submitted to some experimentation and turned up to be the group B (Kim, 2015).

It is necessary to assume a "null hypothesis" in order to work with the T-test. The hypothesis affirms that the means of the groups are equals, and it can be accepted or rejected, which the latter would indicate that the difference between the two groups are solid and probably did not occur by coincidence. The test generates a "t-value" (also called "t-score") where the closer the value is to zero more the similarity between the two group means. The "t-value" for a paired test can be calculated using Equation 6.7. The t-value can be compared to a "T Distribution Table" which helps in rejecting or accepting the null hypothesis, using a parameter called "level of probability" (also known as alpha level or level of significance). The level of probability must be chosen as a criterion to define the acceptable chance of the difference occurred just by coincidence. This value is commonly defined as 5%. The test also outputs the "degrees of freedom," corresponding to the number of values that are free to vary, which in the case of a paired T-test it can be calculated by Equation 6.8.

$$\text{t-value} = \frac{mean1 - mean2}{\sqrt{\frac{var1^2 + var2^2}{2}} * \sqrt{\frac{2}{n}}} \tag{6.7}$$

where n corresponds to the number of samples in each group; the $mean1$ and $mean2$ are average values of each group; and the $var1$ and $var2$ correspond to the respective variances.

$$\text{Degrees of Freedom} = (2n - 2) \tag{6.8}$$

where n corresponds to the number of samples in each group.

The paired t-test is the one chosen for the infusion experimentation in this work. The group A is going to contain a specific number of models without the infusion technique. The group B will contain the same number of models, except they will be trained using the infusion. This will enable the comparison about the effective amount of change before and after the application of the technique.

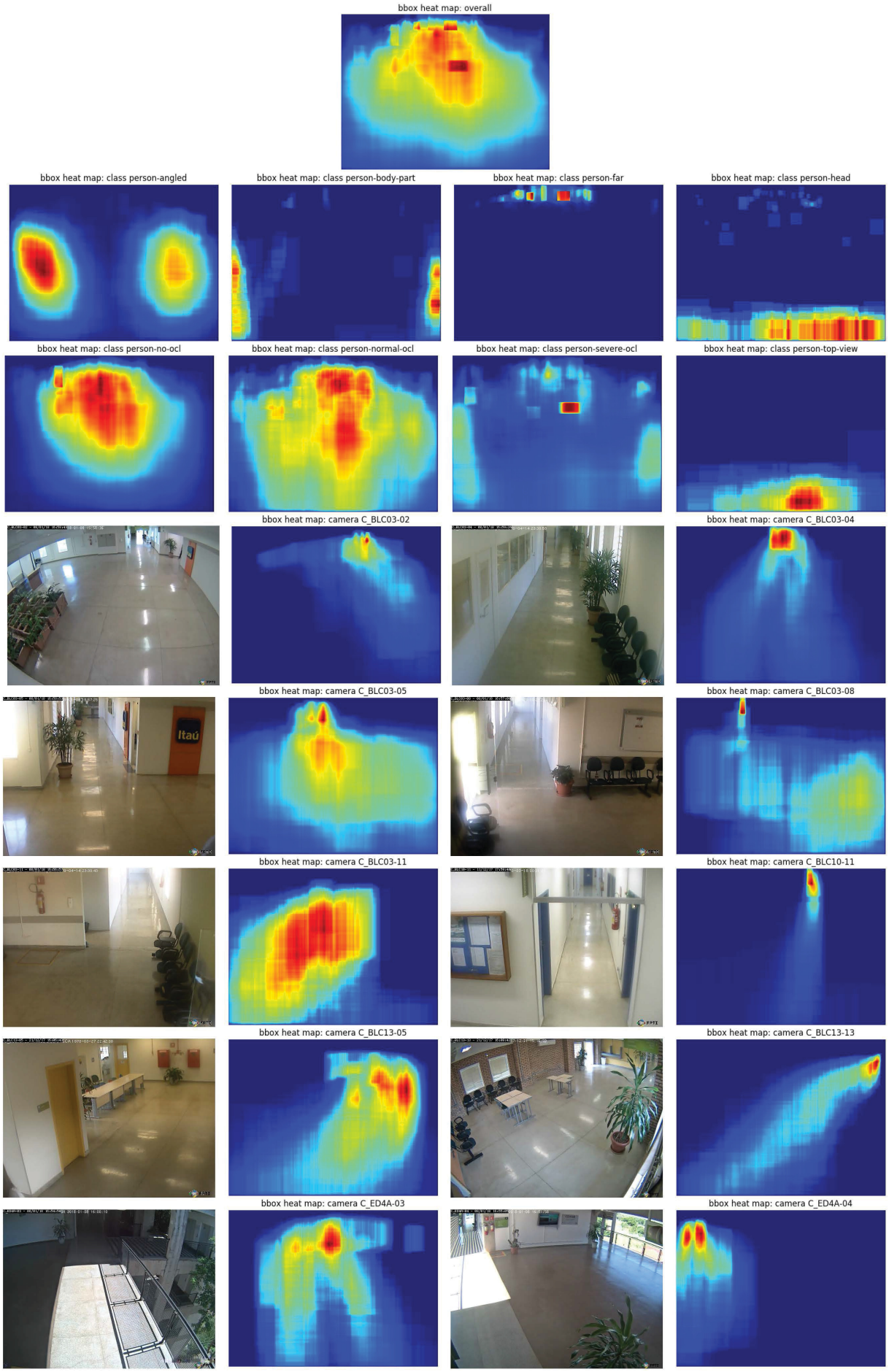


Figure 6.13: PTI01 bboxes heat map in: overall, by camera samples and by classes. On the left of each camera's heat map, the respective scenarios are presented.

7 RESULTS AND DISCUSSION

In this Chapter, we report the main results for all relevant experiments in this work. We follow the general strategy presented in Chapter 5 focusing on evaluating the performance of YOLOv3 on the Caltech Pedestrian Dataset and the PTI01 Pedestrian Dataset. Also the "weak semantic segmentation infusion" is tested in order to check for improvements in the detector's accuracy. The datasets and the metrics have been explained in Chapter 6, as well the training environment. Tables, charts and textual discussion are applied, varying according to each experiment. The comparison of accuracy scores from different sections within this Chapter is discouraged because they are not guaranteed to be using the same configuration and may lead to miss-understandings of the model's performances. The results in each section are well suited for local comparison only. The best results achieved with the YOLOv3 are separated in two sections, one for the default model (7.1.8.2) and the other using the infusion technique (7.2.2.1).

7.1 YOLOV3 EXPLORATION

In this Section, the YOLOv3 model is evaluated through the exploration of many configurations in the experimentation scenario. The experiments were about understanding the influence of the annotations quality of both datasets for the models (Section 7.1.1); the effectiveness of the proposed "canonical bounding boxes" (Section 7.1.2); the differences when using different numbers of anchors produced by distinct methods (Section 7.1.3); the model's ability in detecting each of the eight classes in the third version of PTI01 (Section 7.1.4); the influence of the data augmentation in the training process according to different configurations (Section 7.1.5); the five compositions of the PTI01 dataset planned to have different difficulties (Section 7.1.6); different settings for the mAP metric (Section 7.1.7); the model's performance on three configurations of the Caltech dataset (1×, 10× and sanitized), as well comparing the model to the state-of-the-art and checking some different model parameters (Section 7.1.8); the differences in performance between the YOLOv3 and YOLOv3-tiny models (Section 7.1.9); the model performance regarding speed for different configurations (Section 7.1.10). Lastly, in Section 7.1.11, we briefly summarize the general conclusions about the executed experiments.

7.1.1 Impact of bad quality annotations

According to the dataset descriptions in Chapter 6 and the proposed evaluations in Chapter 5, it is desired to test the YOLOv3 model's behavior when submitted to different qualities of annotations from the datasets PTI01 and Caltech. It is naturally expected that the network would benefit from more precise and coherent labeling, which would provide more meaningful information for the future inferences. We have used the general default training procedures described in Chapter 6 where only the annotations have been changed in the environment. The number of training epochs may vary according to the progress achieved by each model.

Figure 7.1 presents accuracy in training steps for the PTI01 Pedestrian Dataset. In this case, a training step represents an epoch where the model has demonstrated improvements in the "validation loss." Epochs which did achieve lower "loss" performance are excluded. Two versions of the dataset have been tested. The first is "PTI01 version 1" (v1) which has lower annotation quality and the "PTI01 version 3" (v3) corresponding to the last and sanitized version. It is possible to verify that the model achieves lower accuracy in the v1, probably due to more

occurrences of incoherent annotations which confuses the network when learning. Once the annotations are more meaningful, the model demonstrates to have a better knowledge of the data. Another curious fact is that the model, when training over the v3, made use of more training steps (epochs) than the v1. The model in v3 did not stop earlier because it kept having improvements in the validation loss for about four more epochs. We understand that the model has been enabled to extract useful information for a more extended number of iterations instead of being forced to give up earlier.

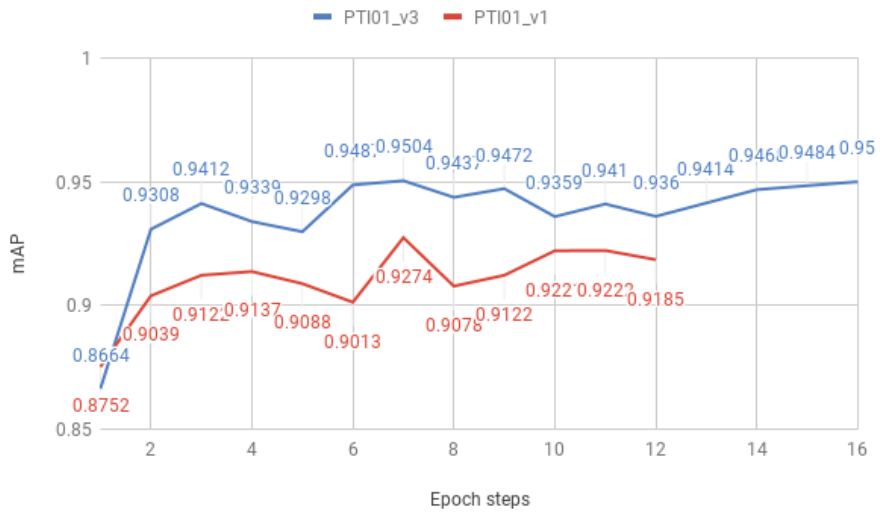


Figure 7.1: Comparison between PTI01 version 1 annotations and version 3 with higher quality annotations. The accuracy is presented in mAP-40. Each epoch step represents a sequential epoch in the training process.

The Caltech Pedestrian Dataset is also evaluated in four different configurations (Figure 7.3). The first is the "Full Original" where the model has been trained with the the original annotations from the "Caltech1×" training set, and tested also over original annotations (testing set). The second configuration ("Tested Sanitized") used the same model trained from the "Full Original", but now tested over the sanitized annotations set provided by Zhang et al. (2016b). In a third configuration ("Full Sanitized"), a new model is trained using the sanitized training set (Zhang et al., 2016b) and tested over the sanitized testing set. Lastly, the "Train Sanitized" uses the same model from "Full Sanitized", but now tests the model with the annotations from the original training set. In all the evaluations, the accuracy has been considered only for the class "Person" which is the primary and most relevant. As expected, the "Full Sanitized" version of the Caltech's annotations (Zhang et al., 2016b) provided a better detection quality to the YOLOv3 model. The results also demonstrate that there an important difference in accuracy when the sanitized annotations are considered as the testing set ("Test Sanitized") compared when it is used in the training set ("Train Sanitized"). The non-sanitized annotations are less harmful to the accuracy when they are used for training instead when using for testing. In other words, seems like the YOLOv3 model executes its own sanitizing during training having similar accuracies between "Full Original" and "Train Sanitized."

The Figure 7.2 shows some examples of bad quality annotations that can be found in the dataset, and are the main reason for the existence of the sanitized version of the annotations.

In the same way, the PTI01 using improved annotations enabled the achievement of greater accuracy. The difference in accuracy between both experiments is around 20% points which we find dramatically expressive, denoting the substantial effect of bad quality annotations

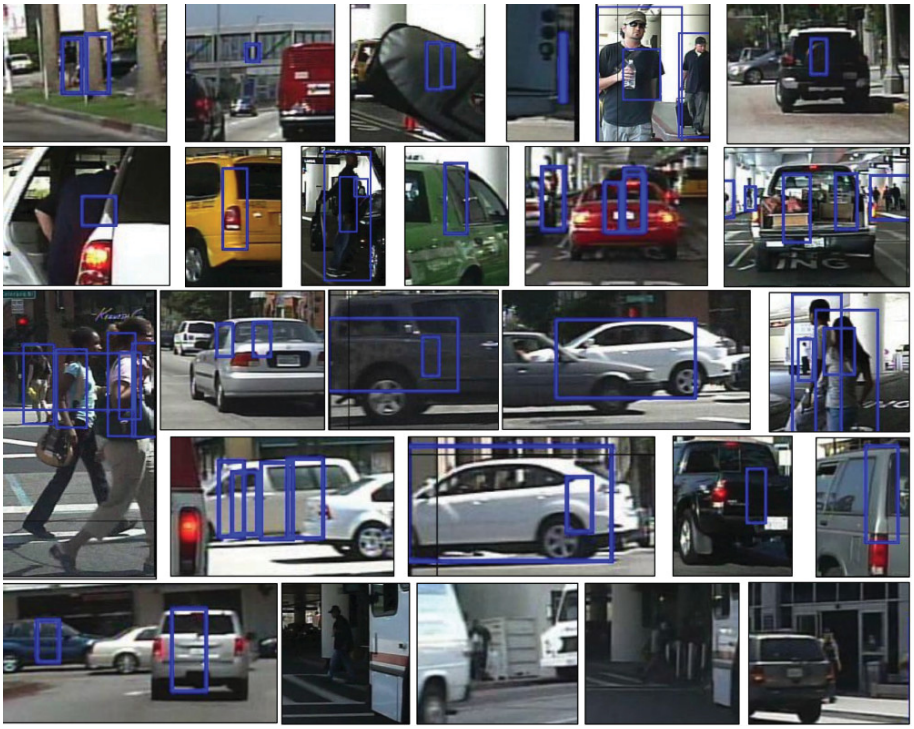


Figure 7.2: Examples of wrong annotations present in the Caltech dataset. The bounding boxes are presented in blue. In each image, it is possible to verify that many annotations wrongly define part of other objects as being pedestrians. In the last four samples, pedestrians have not been annotated at all.

in the YOLOv3 model. Another interesting aspect of the sanitized training is that it did run for almost the double of the epochs, probably trying to explore any additional information.

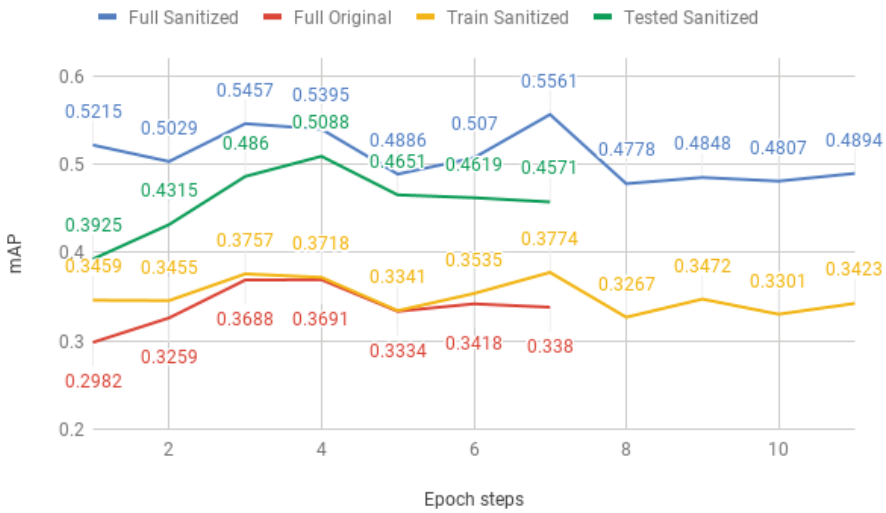


Figure 7.3: Accuracy comparison for the "Caltech1×" dataset configuration between sanitized version of the annotations and original ones. The metric is mAP-40 which enables a easier visualization of the discrepancy between the tests. Configurations: Full Original: trained and tested with original annotations; Full Sanitized: trained and tested with sanitized annotations; Trained Sanitized: trained on sanitized and tested on original; Tested Sanitized: trained on original and tested on sanitized.

7.1.1.1 *Caltech10× and the decreasing accuracy effect*

When experimenting with Caltech10× we saw an interesting effect related to the model's accuracy over training time. In this dataset, contrary to the PTI01, the best accuracies were found in the very initial epochs. Figure 7.4 presents in blue the mAP-50 performance over an increasing number of epochs. It is possible to see that the accuracy starts at some point and only decreases after each epoch. To help in understanding such behavior, two additional models have been trained with the same configuration unless by changing the learning rate to half of the default value, and also to a tenth. The idea was to check if there was any influence of the learning rate to this observed effect. By reducing the learning to half, it was expected to see some change. However, there was no significant variation in the training behavior as can be verified in the linear tendency presented by straight lines in Figure 7.4. When reducing the learning rates, it just makes the accuracy decrease to happen slower.

The Caltech1× and Caltech1× Sanitized also present such behavior but in a milder way. Despite not having an explanation for this effect, we understand that the YOLOv3 model may not efficiently extract useful knowledge from the Caltech datasets to make the current weights better than the initial pre-trained weight which has been trained on the COCO dataset (Lin et al., 2014). It seems like the Caltech dataset data only disturbs the model's state in each epoch.

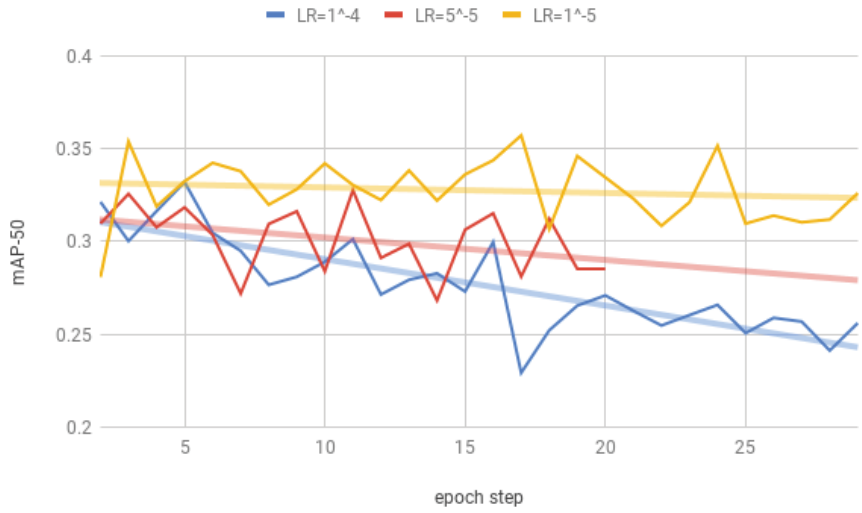


Figure 7.4: Comparison of the accuracy progress of YOLOv3 on Caltech10× according to different learning rates. There is no significant change in the training behavior when using half (red) and a tenth (yellow) of the original learning rate (blue). The straight lines represent the linear tendency of each model, where the lower learning rate demonstrates a slower decrease in accuracy compared to the higher rates. The model with learning rate set in 5^{-5} (red) stopped earlier than the other two models.

7.1.2 Canonical Bounding-boxes

Inspired by Dollár et al. (2009b), which has applied a fixed ratio for the pedestrian bounding boxes in the Caltech dataset, a similar approach is tested over the PTI01 dataset. Instead of using a single ratio value we propose to use multiple standardized ratios to generate canonical bounding boxes. The idea is to adjust every bounding box so that they match aspect ratios in a factor of 1.0. This will guarantee that the relationship between width and height are more stable, producing more regular pedestrian shapes rather than letting every bounding box with non-standardized ratios.

The adjustment algorithm will update the width or the height depending on the aspect ratio. If the ratio is bigger than the maximum ratio allowed (3.0 in this scenario), the width will be increased until the ratio decreases to match the maximum ratio. If the ratio is below the maximum ratio, the height is increased to match the closest allowed aspect ratio. Both operations were projected to never lose information about the pedestrian so that the bounding boxes will always become bigger than its original version.

In this experiment, we test four different configurations related to the canonical bounding boxes. The models are the YOLOv3 and its "tiny" version, while the PTI01 dataset is utilized in the "high bias" setting. Figure 7.5 presents the new distribution of the aspect ratio for the PTI01, which can be compared to the original distribution in Figure 5.6. The bounding box adjustments can also be verified in Figure 7.6 where samples of original and canonical versions are compared.

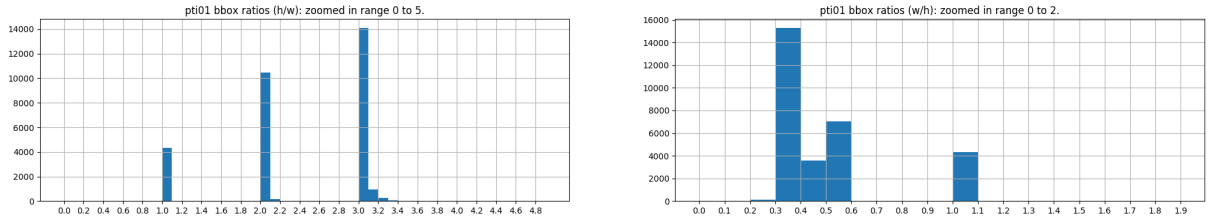


Figure 7.5: The bounding box distribution of the aspect ratios in the PTI01 dataset after the application of the canonical conversion. Left is height/width. Right is width/height. It is possible to see that the bounding boxes have been moved to the standardized ratios of 1.0, 2.0 and 3.0 (in the height/width). Due to the impossibility of setting the exact ratios of the pre-defined values, some boxes have been set to approximated values.

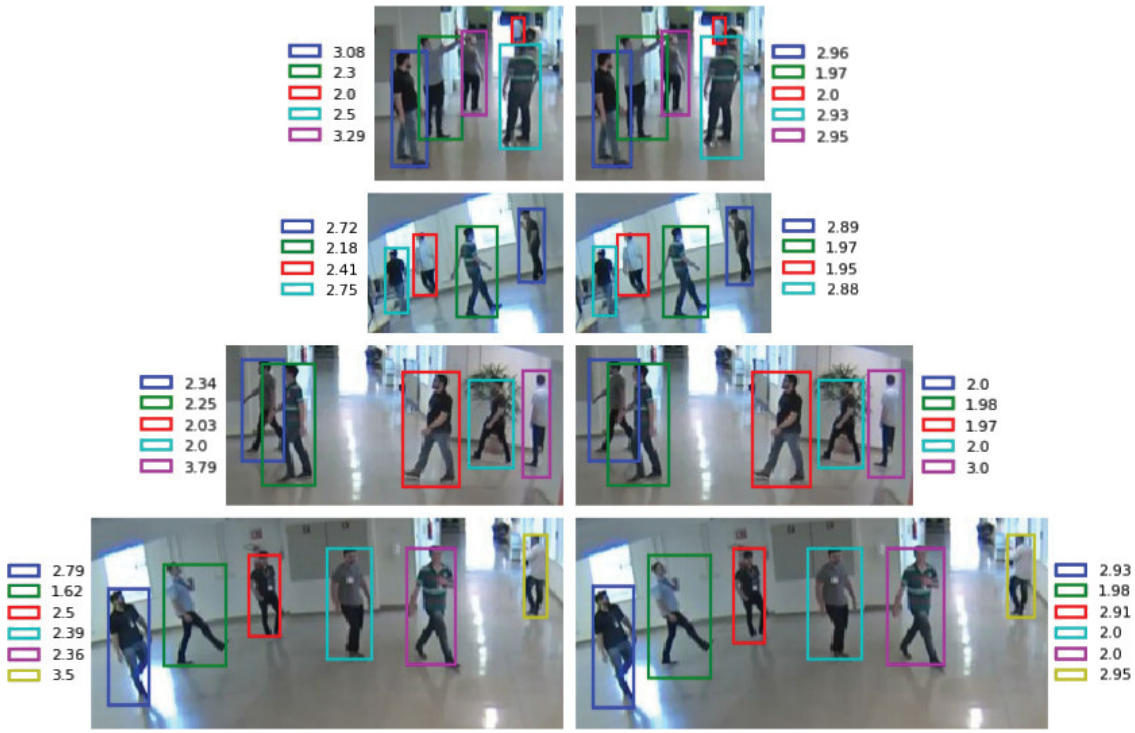


Figure 7.6: Comparison between original (left) and canonical (right) bounding boxes. The aspect ratio values are presented in the sides of each sample. It is possible to verify that the adjustments are very subtle, but, visually, the annotations get in a better fit.

Table 7.1 shows the results. The "GT Training" explains which dataset version was used to train the model: the original (our baseline) or the canonical version where the bounding boxes

have been adjusted to the canonical ratios. After the model is trained, it is possible to evaluate it against the original dataset or over the canonical version, which is described by the column "GT Metric." This variation is proposed for exploratory purposes. Another option when testing is to use the bounding boxes directly from the network output which is the default or adjusting those predicted bounding boxes according to the canonical setting, which is defined in the column "Post-Processing." We propose this configuration in order to check if the network would be able to directly generate predictions "enough canonized," or if it would be mandatory to apply the adjustment to the network's output.

According to the achieved results, there was no configuration able to overcome the baselines in accuracy. Also, it was possible to verify that applying the post-processing did decrease the performance of the predictions, mainly for YOLOv3-tiny. Because of the bad results, the proposed canonical configuration for the aspect ratios demonstrated to be inefficient in this experiment.

Table 7.1: Canonical Bounding Boxes results. Best accuracies belong to the baselines (in bold).

Model	GT Training	GT Metric	Post-Processing	mAP
YOLOv3-tiny	Original	Original	None	**0.8092**
YOLOv3-tiny	Canonical	Original	Canonical	0.7744
YOLOv3-tiny	Canonical	Original	None	0.7921
YOLOv3-tiny	Canonical	Canonical	Canonical	0.7801
YOLOv3-tiny	Canonical	Canonical	None	0.7936
YOLOv3	Original	Original	None	**0.9541**
YOLOv3	Canonical	Original	Canonical	0.9375
YOLOv3	Canonical	Original	None	0.9397
YOLOv3	Canonical	Canonical	Canonical	0.9378
YOLOv3	Canonical	Canonical	None	0.9429

7.1.3 Anchors

A general procedure when training the YOLOv3 model is to recalculate its "anchors." As explained in Chapter 3, the anchors are the default bounding-boxes for each cell within the grid created in the YOLOv3's network calculation structure. The predictions generated by the network are offsets which summed with the default bounding-boxes (anchors) match the wanted object. More similar the anchors are to the general shapes of the dataset's objects, easier will be to the network learn to estimate the correct location and dimensions of the object. Redmon and Farhadi (2018) use a clustering algorithm to find nine groups of the ground truth bounding boxes with a similar shape. Those groups compose the three anchors selected for each one of the three scales. In the YOLOv3-tiny model (a smaller version of the YOLOv3 network) there are six anchors, three for each of the two scales.

In this experiment, we want to compare different configurations for the anchors specifically for the PTI01 dataset. The initial evaluations are executed over the YOLOv3-tiny model. The final experiments are evaluated over the default model.

We have used three main configurations for the anchors. The first is the "default" configuration which corresponds to the original anchors provided by Redmon and Farhadi (2018), wherein the anchors have been originally calculated over the COCO Dataset (Lin et al., 2014). The second is the "re-calculated" configuration where the default algorithm for re-generating the anchors is utilized over the PTI01 dataset. In this configuration, Redmon and Farhadi

(2017) proposed to find good anchors through using the k-means clusterization algorithm with a customized distance metric that takes the IoU into account. The last configuration is composed of a simple implementation of the k-means algorithm[1] applied by the authors of this work where six different quantities of anchors are tested. Table 7.2 presents the results.

The initial hypothesis is that the "re-calculated" version would provide a greater accuracy compared to the "default" configuration due to the fine-tuning over the PTI01 dataset, which could supply better anchors according to the data. However, the "re-calculated" version has shown degradation of almost 20 points in accuracy compared to the default. The reasons behind this bad results remain unsolved in our scenario.

As the default recalculation method has shown unexpected worse results compared to generic anchors, we are encouraged to test another method. We choose to implement a simple k-means algorithm with the default Euclidean distance metric. The result for six anchors achieved similar accuracy to the "default" COCO anchors. This is better than the "re-calculated" anchors but still not better than the generic ones.

In order to evaluate the impact of a different number of anchors per cell, we have experimented from 2 to 12 anchors considering only even numbers once there are two heads in the YOLOv3-tiny network. When changing the number of anchors to 4, 8, 10 and 12, the accuracy did not show significant variations but a slight decrease in higher quantities. Surprisingly, the best accuracy has been achieved by two anchors only, that is, one anchor of each head. The improvement corresponds to almost three percentage points. We understand that the PTI01 dataset, once composed by a single class (by default), would not require too many shapes for the pedestrians compared to the COCO dataset which is composed by 80 classes in many shapes. To check the accuracy improvement, we test the same logic in the YOLOv3 default model. As this model has three heads, the number of anchors is required to be multiple of three. In such way, the minimum number of anchors are three, one for each scale (Figure 7.7). In Table 7.3 four different epochs are compared using the default nine anchors originally calculated over the COCO dataset (Lin et al., 2014) and our k-means re-calculation with three anchors. It is possible to see there is a slight improvement in accuracy (about 1%) when using our method, which confirms the benefit of defining the number of anchors in the minimum for the PTI01 dataset. Additionally, our k-means implementation demonstrates improvements in using three anchors instead of nine over the Caltech10× dataset (Table 7.4).

Table 7.2: YOLOv3-tiny anchors experiment on PTI01.

Configuration	Default	Re-calc.	Our k-means					
Anchors (k)	6	6	**2**	4	6	8	10	12
Score	0.8214	0.6372	**0.8478**	0.8232	0.8202	0.8180	0.8106	0.8175

Table 7.3: YOLOv3 anchors experiment on PTI01.

Anchors	**Score from the last epochs**			
YOLOv3 (default 9 anchors)	0.9419	**0.9423**	0.9415	0.9419
YOLOv3 (our k-means 3 anchors)	**0.9541**	0.9520	0.9507	0.9525

[1]https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Table 7.4: YOLOv3 anchors experiment on Caltech10×. The score uses the Reasonable metric in LAMR, where the lower the better.

Configuration	Our k-means	
Anchors (k)	**3**	9
Score (Reasonable)	**0.1858**	0.2273

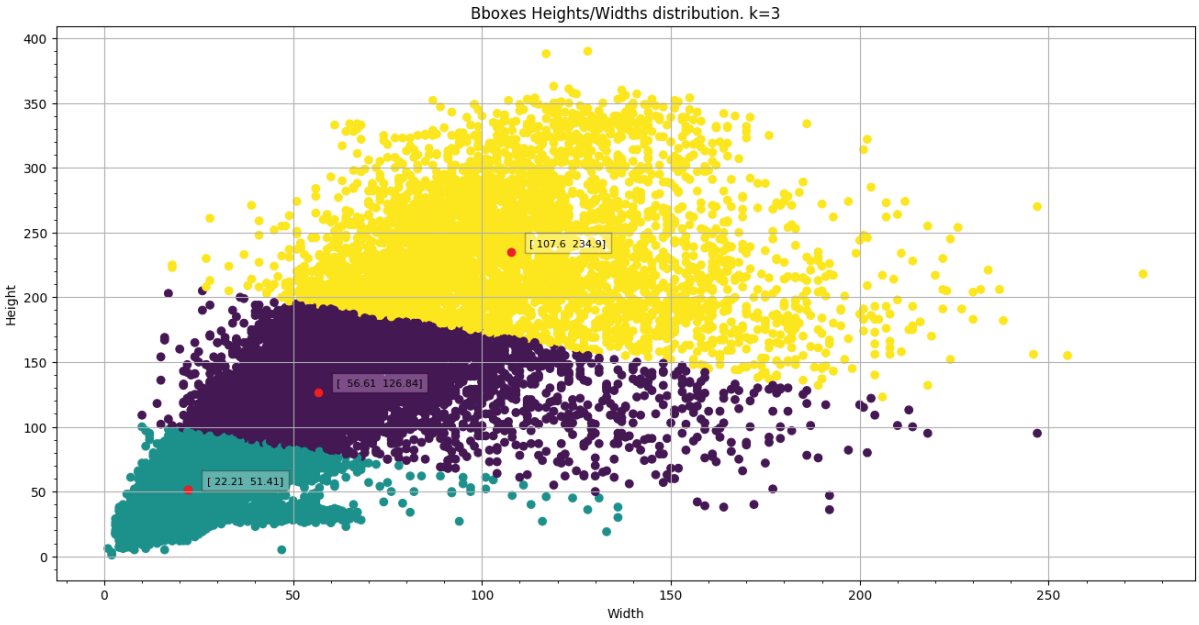


Figure 7.7: Bounding boxes distributions in width and height for the PTI01 dataset, and the clustering example. Each dot corresponds to ground truths. The three colors are relative to the three groups defined by the k-means algorithm, wherein the three dots in red are the "centroids" of each cluster, that is, the anchors themselves.

7.1.4 Extra classes

Inspired by Liu et al. (2018) and Dollár et al. (2012), in this experiment, additional classes for the PTI01 are evaluated in order to try understanding the intra-class appearance variations for the pedestrians. As proposed in Chapter 5, eight classes have been created, and each one is connected to pedestrians in the scene but are grouped according to the similarity in the general appearance. The models are trained over the easiest configuration of the PTI01 dataset, named "high-bias" (description in Chapter 6).

In some of the experiments, classes are merged or excluded from the dataset in order to understand the influence of such configurations. For instance, it was important to verify the hardest class among the eight ones and, after that, what would be the impact in the model's performance if we remove those annotations from the dataset. We naturally expect to achieve greater performance in the model once the annotations of high difficulty are not disturbing the learning process anymore. Another example would be to get two or more classes and put them together, trying to understand whether it was better to keep the appearance variations in different groups (classes) or whether it would be easier to the model keeping the information in a single class.

When classes are merged, the training and predicting follows the new configuration, that is the new set of classes. However, while evaluating the results, we check every annotation for its original class (before merging), so that it becomes possible to understand the class performance

before and after the merge. We call this method as “detranslation mechanism,” and the merging or removal of classes as “translation mechanism.”

The first experiments are elaborated over the YOLOv3-tiny using one baseline and six other configurations. The baseline is a model trained with all the eight classes, and the other six models are trained over different class configurations (merging or removing). The baseline and the other six experiments (A to F) are corresponding to the columns in Table 7.5.

In the table (7.5), the Class Group (CG) is an arbitrary number that defines the merging among the classes, that is, classes with the same CG number are merged. Classes which have been removed are marked as “-”. The TP can be a positive or negative value and represents the number of True Positives increased or decreased compared to the baseline. FP is similar to TP but refers to the False Positives which have been erroneously predicted multiple times to same ground truth in the dataset. It is important to inform that the False Positives which have completely missed the ground-truths are not presented in Table, due to the impossibility of calculating them via the detranslation mechanism.

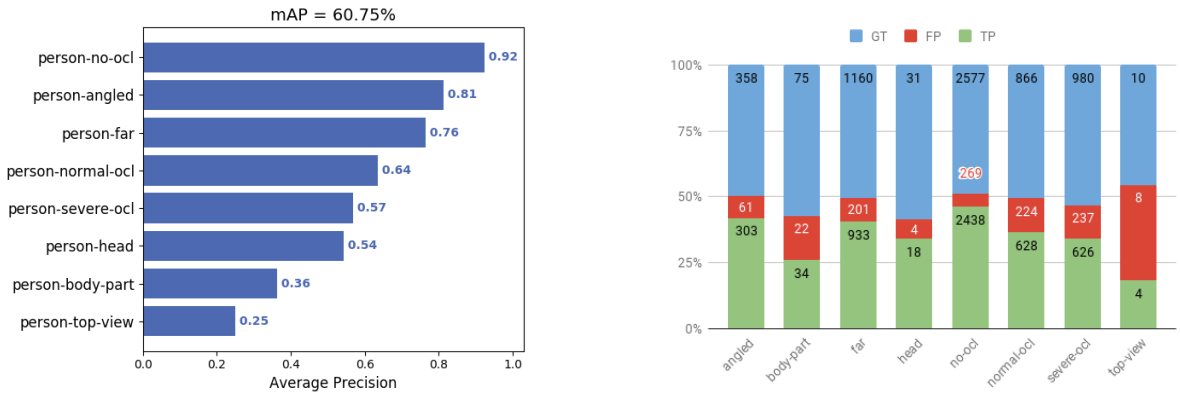
Table 7.5: YOLOv3-tiny trained with extra classes in different configurations (A-F). Each class is related to a Class Group (CG), a True Positive Difference (TP), a False Positive by Multiple Detections Difference (FP), and an Average Precision score (AP).

Classes	**Baseline**		**A**				**B**				**C**			
	CG	**AP**	**CG**	**TP**	**FP**	**AP**	**CG**	**TP**	**FP**	**AP**	**CG**	**TP**	**FP**	**AP**
angled	1	0.81	1	+43	0	0.88	1	+19	0	0.76	-	-	-	
body-part	2	0.36	1	+9	0	0.88	2	+9	0	0.44	-	-	-	
far	3	0.76	1	+23	-7	0.88	-	-	-		-	-	-	
head	4	0.54	1	+3	0	0.88	2	+2	0	0.44	-	-	-	
no-ocl	5	0.92	1	+86	+3	0.88	3	-13	-5	0.92	1	-10	-6	0.92
normal-ocl	6	0.64	1	+162	+3	0.88	1	+100	+5	0.76	2	-21	-5	0.63
severe-ocl	7	0.57	1	+89	-3	0.88	1	+94	+1	0.76	3	+10	-7	0.58
top-view	8	0.25	1	+3	-	0.88	2	+2	-	0.44	-	-	-	

	D				**E**				**F**			
	CG	**TP**	**FP**	**AP**	**CG**	**TP**	**FP**	**AP**	**CG**	**TP**	**FP**	**AP**
angled	1	-5	0	0.79	1	+8	0	0.83	1	+39	0	0.88
body-part	-	-	-		-	-	-		-	-	-	
far	-	-	-		2	+8	0	0.78	1	+29	-6	0.88
head	-	-	-		-	-	-		-	-	-	
no-ocl	2	-3	-8	0.92	3	+7	-4	0.92	1	+77	-7	0.88
normal-ocl	3	-22	-6	0.62	4	-4	-4	0.64	1	+169	+6	0.88
severe-ocl	4	+49	-6	0.62	5	+8	-8	0.58	1	+83	-9	0.88
top-view	-	-	-		-	-	-		-	-	-	

The first conclusion can be taken when analyzing the Baseline AP, where is possible to see the class “top-view” as the one with the lowest score. This result confirms the annotations of pedestrians in the top-view angle represent the most difficult pedestrians to be detected in the PTI01 dataset. That is probably due to the notorious aggressive change in the pedestrian’s appearance. The same behavior is observed in the classes “body-part” and “head,” which are the next with lower scores. As expected, the non-occluded pedestrians (“no-ocl”) are the easiest ones to be detected due to a clear and stable visual. The pedestrians annotated as being “far” are surprisingly well ranked, getting behind only of non-occluded and angled pedestrians. It seems like for the YOLOv3-tiny it is easier to handle small-scale pedestrians than dealing with moderated and severe occlusions. The AP ranking can be verified in Figure 7.8(a), and the relation between True Positives and False Positives is demonstrated in Figure 7.8(b).

The other six class configurations (A to F) are explained below. An additional Table (7.6) reports the overall scores for all classes of each model wherein we focus in the “wAP” metric. The configurations are:



(a) Average Precision

(b) TP × FP × GT

Figure 7.8: YOLOv3-tiny baseline comparative results among the eight classes in the PTI01 dataset, evaluated with the "high-bias" set configuration.

- A: The eight original classes are merged in a single class, that is, the model handles all annotations just as pedestrians. The objective is to compare the model's performance when training with the eight classes separately;
- B: The occlusion classes are merged with the "angled" one. In this configuration, the easiest class "no-ocl," and the most appearance discrepant ones are left independently, while the others with moderated appearance changes are merged. The "far" class is removed as being suspect of negatively influencing the overall performance.
- C: All classes are removed but the non-occluded and occluded ones. Only three classes remain.
- D: Same configuration as C but including the "angled" class. The classes "far," "body-part," "head," and "top-view" are excluded. This is an incremental variation of the previous configurations;
- E: Same configuration as D but including the "far" class. The classes "body-part," "head" and "top-view" are excluded. Another incremental variation of the previous configurations;
- F: Same configuration as E but merging the remaining five classes. The classes "body-part," "head" and "top-view" are excluded;

Table 7.6: YOLOv3-tiny trained with extra classes in different configurations, comparing the overall scores.

Config	mAP	wAP	GR
Baseline	**0.6075**	**0.7779**	**0.8228**
A	0.8836	0.8836	0.8919
B	0.7073	0.8364	0.8707
C	0.7078	0.7854	0.83
D	0.7416	0.7987	0.8396
E	0.7517	0.7931	0.834
F	0.8873	0.8873	0.8963

The main conclusion for the various configurations explored is that merging the classes in a single one makes the model to obtain a better performance. The configuration A and F demonstrate such behavior, wherein the latter achieves a slightly better in the AP due to the removal of the hardest classes. The improvement when excluding those classes is low because the number of samples corresponds to just 2% of the total number of bounding boxes. We understand that it is not essential to remove those classes which are naturally composed of low sampling and higher difficulty of detection. It would be worth removing the classes only if they are not relevant for detection and corresponding to a more significant volume of samples.

YOLOv3-tiny demonstrated that the intra-class variations are well handled for moderated changes in appearance, but suffers for drastic changes. Separating the pedestrians in many classes did not help in improving the accuracy of the more important classes. Contrary to that, when putting the classes all-together, they have increased the TP rate (Table 7.5, experiments A and F). As a collateral effect, merging the classes did show a very little increase in the False Positive rate for multiple detections of the same target, but we judge it as a too small value to be considered relevant.

Also, it was possible to verify that, in the "high bias" setting, the class "far" is not so difficult as naturally supposed and does not provide significant influence in the score when removed from the training set. That situation is also demonstrated by the previous analysis of the baseline, where the class reached the third best score.

Finally, some of the explored configurations are also evaluated in the YOLOv3 default model (Tables 7.7 and 7.8). The baseline scores are reported in Figure 7.9. Observing the results, the conclusions for the extra classes remain the same as for the YOLOv3-tiny experiments. It is evident that in the PTI01 scenario, it is better to work with a single class instead of training the model with multiple pedestrian classes created by intra-class appearance variations.

Table 7.7: Default YOLOv3 trained with extra classes in different configurations (A,B,E and F).

Classes	**Baseline**		**A**				**B**			
	CG	**AP**	**CG**	**TP**	**FP**	**AP**	**CG**	**TP**	**FP**	**AP**
angled	1	0.84	1	+33	-	0.95	1	+10	-	0.86
body-part	2	0.65	1	+5	-	0.95	2	-	-	0.63
far	3	0.91	1	+16	+14	0.95	-	-	-	-
head	4	0.60	1	+3	-	0.95	2	+2	-	0.63
no-ocl	5	0.96	1	+67	+4	0.95	3	-13	-	0.95
normal-ocl	6	0.75	1	+136	+8	0.95	1	+86	+5	0.86
severe-ocl	7	0.75	1	+33	-	0.95	1	+33	+1	0.86
top-view	8	0.51	1	+3	-	0.95	2	+1	-	0.63

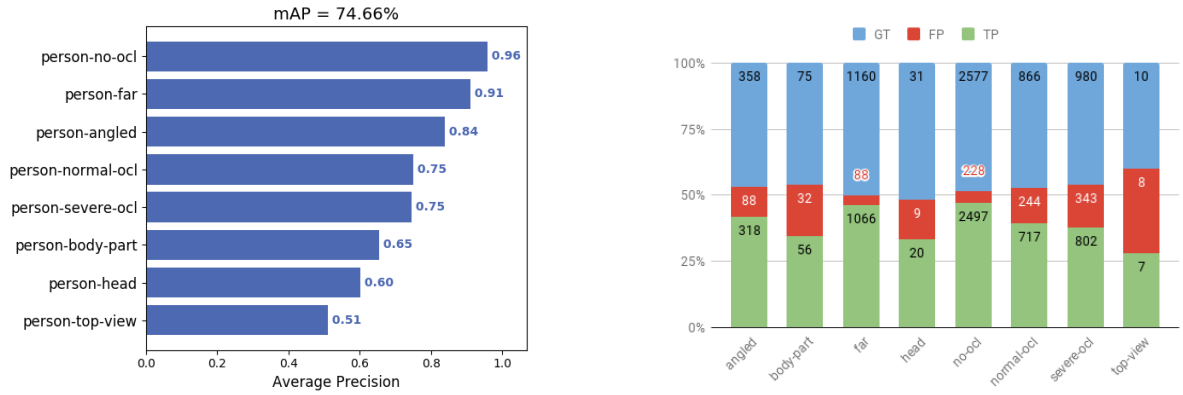
	E				**F**			
	CG	**TP**	**FP**	**AP**	**CG**	**TP**	**FP**	**AP**
angled	1	+6	-	0.86	1	+34	-	0.95
body-part	-	-	-	-	-	-	-	-
far	2	-1	+2	0.91	1	+21	+3	0.95
head	-	-	-	-	-	-	-	-
no-ocl	3	+2	-1	0.96	1	+70	-	0.95
normal-ocl	4	+9	-	0.76	1	+133	-	0.95
severe-ocl	5	+7	+5	0.76	1	+27	-5	0.95
top-view	-	-	-	-	-	-	-	-

However, comparing the baselines, it is possible to verify that the class "far" has achieved the second place in accuracy compared to the YOLOv3-tiny. In the "high bias" setting the scale of the object demonstrated to be less problematic regarding mAP than the occlusions, angling and severe changes in appearance. The lowest scores are still sustained by the last three classes "body-part," "head" and "top-view." Curiously, for the YOLOv3 the normal and severe occlusion obtain similar performance and denotes again that there is plenty of room for improvements in detection of occluded pedestrians. In Section 7.1.9, both YOLOv3 and YOLOv3-tiny models

are evaluated in a harder scenario. In that experiments, the YOLOv3-tiny will demonstrate to have worse performance on the class "far" compared to the default model. The conclusion in the current section about the easiness in handling the small-scaled pedestrian class is probably sustained by the high biases sampling in the training dataset.

Table 7.8: Default YOLOv3 trained with extra classes in different configurations, comparing the overall scores.

Config	mAP	wAP	GR
Baseline	**0.7466**	**0.8722**	**0.9052**
A	0.9504	0.9504	0.9541
B	0.8153	0.9057	0.9263
E	0.8492	0.8809	0.9128
F	0.9546	0.9546	0.9569



(a) Average Precision

(b) TP × FP × GT

Figure 7.9: Default YOLOv3 baseline comparative results among the eight classes in the PTI01 dataset, evaluated with the "high-bias" set configuration.

7.1.5 Data Augmentation

The YOLOv3 (Redmon and Farhadi, 2018) applies a data augmentation technique in order to increase the dataset diversity concerning object's appearances. In the current experiment, three different configurations of the data augmentation are evaluated (Figure 7.10). The first consists of disabling the technique. The second uses the default method provided in the YOLOv3 algorithm which applies about five different modification procedures for each image. The last applies a single operation which randomly decides to horizontally "flip" the training image.

The experiments have been elaborated using the YOLOv3 and the YOLOv3-tiny, taking notes of how many epochs have been necessary to complete the training, and reporting the scores as mAP. The results are presented in Table 7.9.

The YOLOv3-tiny demonstrated to be more benefited from the "flipping only" data augmentation technique. Surprisingly, the default augmentation method decreases the model's performances. This may be related to the network's low complexity, which probably has a higher difficulty to understand the appearance variations provided by the default augmentation method. It seems to be so hard to deal with such data augmentation that not using it enabled better performance. The configuration for just flipping the images horizontally ("flipping only") has shown the best results. As this data augmentation do not cause drastic changes in appearance,

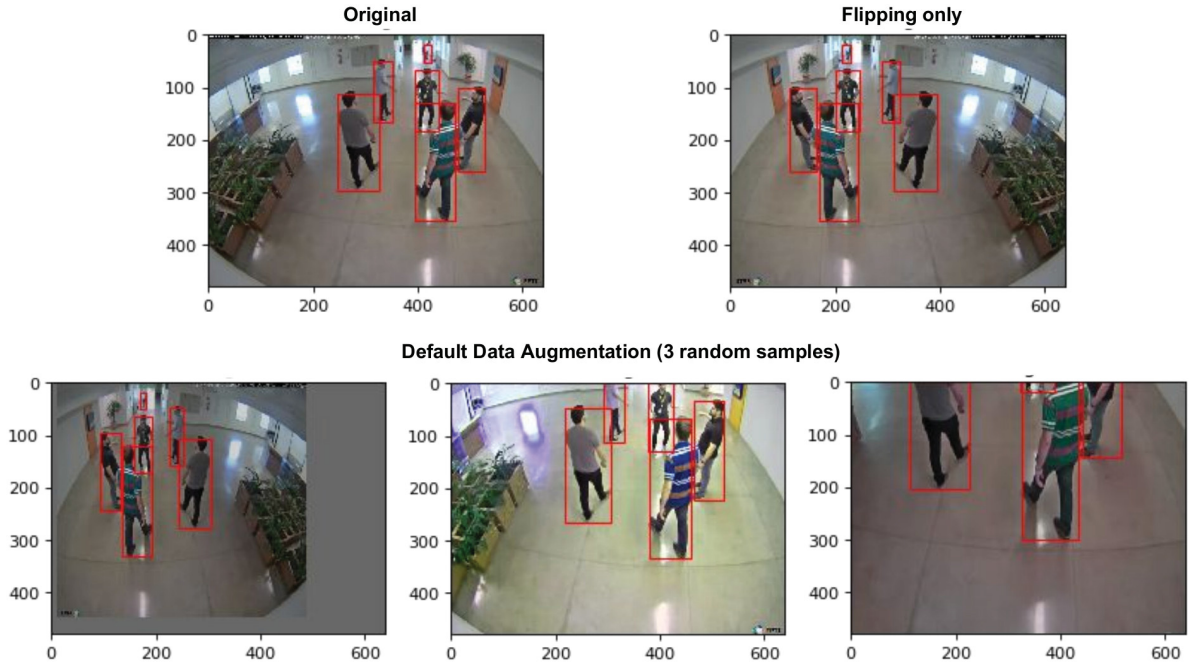


Figure 7.10: Comparison of two data augmentation methods and the original image. The default data augmentation provided by the YOLOv3 randomly applies a set of modifications in the original image (3 samples in the bottom line). It is possible to see (bottom-right image) that the default data augmentation method does synthetically create pedestrian occlusions.

Table 7.9: YOLOv3 and YOLOv3-tiny models tested with three different data augmentation configurations on PTI01 dataset.

Model	Data Augmentation	Epochs	mAP
YOLOv3-tiny	None	5	0.8649
YOLOv3-tiny	Default	32	0.8478
YOLOv3-tiny	Flipping only	9	**0.8812**
YOLOv3	None	17	0.9458
YOLOv3	Default	49	**0.9541**
YOLOv3	Flipping only	18	0.9484

the “tiny” model did make progress in acquiring useful knowledge. It is also interesting to check the number of epochs necessary to train in each configuration. Not using data augmentation lead the model to achieve great accuracy in just five epochs. Adding the flipping only method, it almost doubled (9 epochs), but still being a low quantity compared to the default augmentation which required 32 epochs.

While observing the default data augmentation, we have noticed that some of the randomly generated samples would produce artificial heavy occlusions. The bottom-right sample from Figure 7.10 exemplifies such situation. The pedestrians lose a great quantity of the body information. In our opinion, this is one of the reasons the training with the default data augmentation takes many more epochs to complete, probably due to a higher difficulty in understanding the pedestrian appearance which varies too much in that case. Also, it may be related to the YOLOv3-tiny lower performance when training with it.

Table 7.10 enables the comparison between versions of YOLOv3 model using the default data augmentation and the “flipping only method.” Observing the Caltech results the default data augmentation method improves the accuracy for the “reasonable” evaluation and the

non-occluded pedestrians, but a decrease in the occlusions cases. In the PTI01 dataset the same behavior is observed, but with a slightly lower score variation.

Table 7.10: Influence of data augmentation on occlusions. Comparison of the YOLOv3 model trained with the default and "flipping only" data augmentation methods and the accuracy scores obtained in different levels of occlusion for the datasets Caltech1× Sanitized (Zhang et al., 2016b) and the PTI01 with the "leaving events out" configuration. The default data augmentation seems to prejudice the accuracy of occluded pedestrians. For LAMR, the lower the score, the better. For the mAP-40, the higher the score, the better. The results for the PTI01 dataset are defined by the pedestrian classes "no-ocl," "severe-ocl" and "normal-ocl," respectively. Best scores are in bold.

		LAMR			
Dataset	**Data Aug.**	**Reasonable**	**No Occlusion**	**Heavy Occlusion**	**Partial Occlusion**
Caltech	Flipping only	0.2019	0.1805	**0.5433**	**0.3491**
	Default	**0.1801**	**0.1530**	0.6075	0.3602
		wAP-40	**mAP-40**		
Dataset	**Data Aug.**	**All Classes**	**No Occlusion**	**Severe Occlusion**	**Normal Occlusion**
PTI01	Flipping only	**0.7041**	0.8790	**0.3786**	**0.5075**
	Default	0.6950	**0.8884**	0.3407	0.5025

Interestingly, the metric wAP-40 which evaluates all the classes at once demonstrates that for the PTI01 dataset the "flipping only" method enables the best overall score. This confirms that if one is focusing on the detection of occluded pedestrians, the YOLOv3's defaults data augmentation method are going to reduce the potential of the model compared to simpler techniques like the "flipping only." As a complementary analysis, Table 7.11 presents the scores for the other classes from the PTI01 dataset comparing both discussed data augmentation techniques. Curiously, the default method did negatively impact the detection of classes "Head," "Top view," and slightly the class "Far." However, it did positively impact the learning on the "Angled" class, while having similar results for the "Body part."

Table 7.11: Influence of data augmentation on additional classes of PTI01 dataset. This is complementary to Table 7.10.

		mAP-40				
Dataset	**Data Aug.**	**Angled**	**Body part**	**Far**	**Head**	**Top view**
PTI01	Flipping only	0.4877	0.2106	**0.7170**	**0.2000**	**0.3326**
	Default	**0.6005**	**0.2128**	0.7091	0.0007	0.0006

For the YOLOv3 default model, the three augmentation configurations have provided similar scores, varying less than 1%. The difference from the flipping only method to the default one is about 0.6%, at the cost of 31 additional epochs. According to the achieved results, we understand that while training the YOLOv3 for general experimentation, it would be beneficial to use the "flipping only" data augmentation method, wherein the convergence is relatively faster with similar accuracy. However, to achieve the maximum performance, it would be necessary to spend the double of epochs to increase the accuracy in half percent. Nevertheless, we also suppose that maybe changing some less aggressive parameters in the data augmentation, such as color saturation, it could provide a comparable increase in performance as the default model, but without requiring so many epochs.

In order to check the "flipping only" versus default performances in a different environment, an evaluation over the Caltech dataset has been executed. The model utilized is the default YOLOv3, and the annotations for the dataset are the sanitized ones given by Zhang et al. (2016b). According to the achieved results (Table 7.12), it was possible to verify that the behavior with the "flipping only" and default data augmentations is similar between the Caltech and PTI01 datasets. However, there was an increase of almost 1% in accuracy when using the simpler

method (“flipping”) which resembles that the default data augmentation happens to slightly disturb the training. This could be related to the fact that the default data augmentation process may be hardening too much the scenes that are already more difficult than the PTI01. Again, using the “flipping” method reduced the number of needed epochs significantly to reach the final scores, requiring only about 37% of the iterations (2.7× less).

Table 7.12: YOLOv3 model tested with two different types of data augmentation on the Caltech dataset with the sanitized annotations provided by Zhang et al. (2016b).

Model	**Data Augmentation**	**Epochs**	**mAP**
YOLOv3	Default	35	0.5479
YOLOv3	Flipping only	13	0.5561

An additional experiment has been elaborated as a variation to the data augmentation methods already presented in this section. The objective was to increase the network’s input size to the double of the original size, and check if there were any improvements due to the resolution mechanics within the network. This test was executed over the PTI01 dataset where both the images and the network input used by default the resolution of 640×480 pixels, and without any other data augmentation. For this micro-experiment, we changed the image and the network input to 1280×960. This model achieved an mAP of 0.9443 which is about the same score compared to the 0.9458 in the YOLOv3 with no data augmentation (Table 7.9). We find that increasing the network resolution without having an image originally of a bigger size does not offer any significant improvement in the final accuracy.

7.1.6 PTI01 in different configurations

In this experiment, the main configurations for the PTI01 dataset are evaluated using the YOLOv3 and the YOLOv3-tiny models. Additionally, secondary experiments are executed for complementary analysis.

Table 7.13 summarizes the four dataset configurations described in Section 6.4.2, and an extra one called “High bias & discarding frames” which discards 66% of the training frames in the high bias setting. The metrics mAP (IoU threshold in 40%) and Log Average Miss Rate are reported. Figure 7.11 presents the Miss rate × FPPI curve for the experiments.

Table 7.13: Results for multiple configurations on PTI01 dataset for the YOLOv3 and YOLOv3-tiny models. For mAP, the higher the better. For the Log Average Miss Rate, the lower, the better.

Experiment	**mAP-40**		**Log Average Miss Rate**	
	YOLOv3	YOLOv3-tiny	YOLOv3	YOLOv3-tiny
High bias	0.9504	0.8812	0.0936	0.2127
Leaving events out	0.9084	0.7672	0.1591	0.4083
Leaving 1 camera out	0.8494	0.6834	0.1956	0.3678
Leaving 3 cameras out	0.8309	0.7235	0.2499	0.4033
Leaving 5 cameras out	0.8177	0.7085	0.2561	0.4222
Leaving 3 cameras out & discarding frames	0.8184	0.6646	0.2608	0.4945
High bias & discarding frames	0.9306	0.8331	0.1162	0.3005

Analyzing the results, it is possible to verify that YOLOv3 and YOLOv3-tiny sustain similar behavior to the different dataset configurations, independently of checking the mAP or Miss Rate. For both models, the easiest setting is the “high bias,” which is expected due to the high similarity between the train and test sets. Each camera event had 80% of its frames available for training, while the remaining was used for testing. In such way the model was able to acquire knowledge on scenarios very similar to the ones it has been submitted for testing, representing a

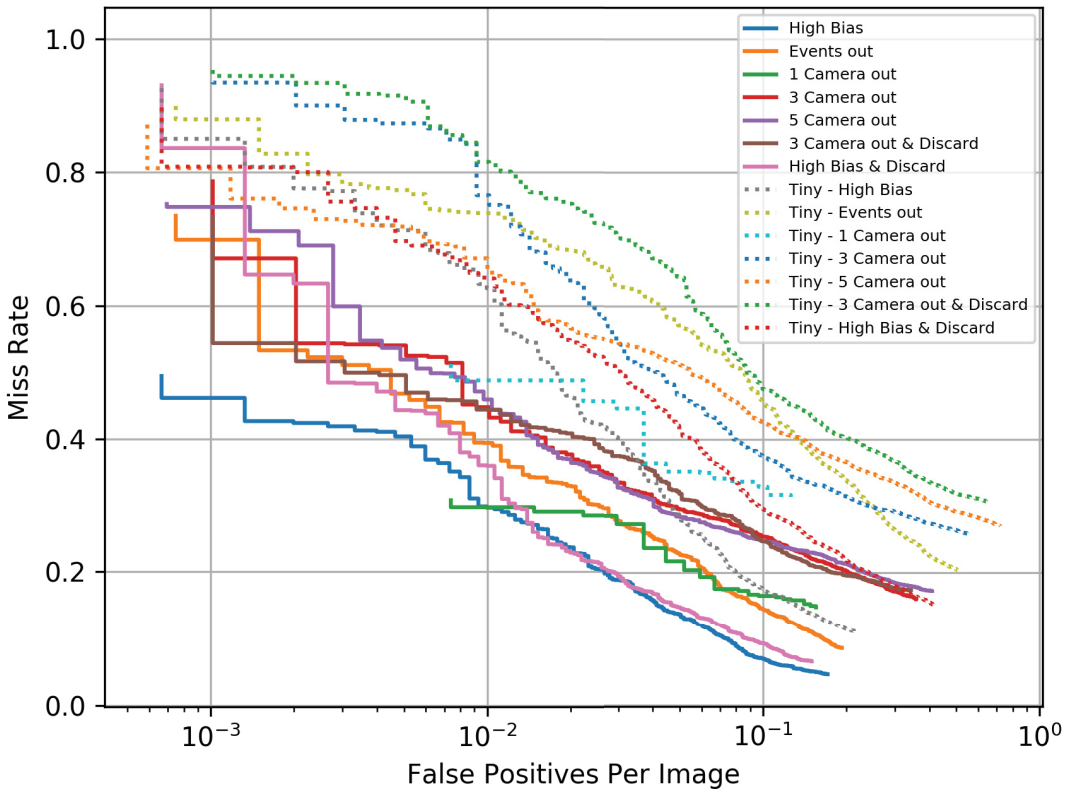


Figure 7.11: Miss Rate × FPPI curve for the different configurations on the PTI01 dataset, using both YOLOv3 and YOLOv3-tiny models.

not so challenging task. This configuration demonstrated to be easy for the YOLO's models so that the YOLOv3-tiny model could achieve impressive accuracy, getting only 7% points behind the complete model in mAP.

For the YOLOv3, the "leaving cameras out" configuration did represent the hardest tests in the evaluation. The models were deprived of 1, 3 or 5 cameras while training, and required to detect pedestrians in cameras never seen before. As explained in Chapter 6, those cameras have been chosen as the most discrepant ones according to some selected metrics. Because of that, it was expected that the model would have a more difficult challenge compared to the "high bias" one. The decrease in accuracy varied from 10 to 15 percent in mAP, depending on the number of cameras left out for testing. While the YOLOv3 model presented a decreasing accuracy according to the increase in the number of cameras left out, the YOLOv3-tiny model demonstrated to suffer more in detecting pedestrians when the most distinct camera was left out than when the 3 or 5 most distinct ones have been selected for the same task. More information comparing YOLOv3 and YOLOv3-tiny is given in Section 7.1.9.

In order to evaluate the model's behavior when there is a decrease in the training data available, two experiments are elaborated (one planned and the other as complementary). The first, "Leaving three cameras out & discarding frames" uses the same scenario of leaving the three most distinct cameras for testing-only purpose while discarding $\frac{2}{3}$ of the training data (frames). In this way, it was expected that the hard task of leaving cameras out would be drastically more difficult when discarding most of the training data. However, the YOLOv3 model achieved similar results even with the lack of data, decreasing the accuracy of less than 3% of the mAP. In

the same situation, the YOLOv3-tiny reduced its performance by 6%. The same occurs for the "High bias & discarding frames" experiment, which suffered a loss of only about 2% with the same discarding method, while the YOLOv3-tiny decreased by 5%.

This experiment demonstrated that even with $\frac{1}{3}$ of the training data, the YOLOv3 model could efficiently extract knowledge from the dataset and achieve similar accuracies compared to a richer dataset. That is proven by, at least, three facts: 1) The PTI01 dataset originally has scenes in low FPS (about 2.5) meaning that there is not much redundant data per second which reduces the amount of information available to the model; 2) Even with such low FPS, $\frac{2}{3}$ of the data are discarded making it three times less data-rich; 3) Three of the most distinct cameras are left out for testing-only, where the model has never been able to learn in such scenario; Such configurations put the model in proof and it still able to sustain similar performances. Nevertheless, YOLOv3 also has shown to take advantage of any extra data to increase the detection quality. YOLOv3-tiny also share the same characteristic, just demonstrating to be more sensitive to the lack of data but still sustaining impressive accuracy.

Lastly, the "Leaving events out" has revealed to be the intermediate configuration of the PTI01 dataset for both default and tiny model versions. Such a result puts this as the main configuration in the current work.

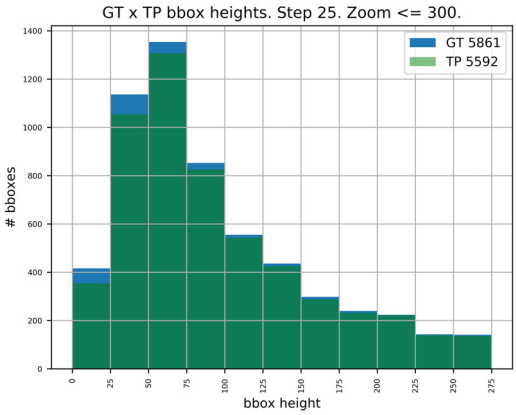
Similar to the "person-far" class evaluated in Section 7.1.4, we run an additional experiment to check the influence of small-scaled bounding-boxes in the model performance. In the previous experiment, the "person-far" class was related to pedestrians with unspecified maximum height. In other words, the annotators (people who did label the dataset) could qualitatively decide whether the person was far from the camera or not. This means that not every single small bounding box in the dataset is related to a small-scaled pedestrian, once it can refer to some other category like a "person's head" close to the camera.

In this experiment, we want to remove any bounding box below a certain height, no matter the class, while the idea is to check their influence in the learning process. Because of the lower performance achieved by YOLOv3-tiny for small-scaled pedestrians compared to the YOLOv3 default model (Section 7.1.4), we choose to execute this experiment only using the default one. Four different configurations are proposed, which they correspond to removing all bounding boxes from the dataset bellow 100, 75, 50 and 25 pixels. The results are reported in Table 7.14. It is possible to verify that the number of bounding boxes decreases significantly according to the increase in the minimum height restriction. About $\frac{2}{3}$ of the bounding boxes are eliminated by the 100 pixels restriction, meaning that the majority of the pedestrians are bellow such height. The mAP variations among the four experiments in relation to the baseline stay in a range of 1%. Despite the elevated quantities of bounding boxes removed in each experiment the reflection in accuracy demonstrated to be not so important. In the baseline, the difference between the IoU-40 (easier) and the IoU-50 (harder) metrics was of 1%. As the height restriction value increased, the difference between both metrics decreased until reaching just 0.37%, demonstrating that the model tends to provide predicted bounding boxes of lower quality when detecting small-scaled pedestrians. No other relevant facts have been noticed, and we find that the model does not benefit from removing small-scaled pedestrians from the dataset.

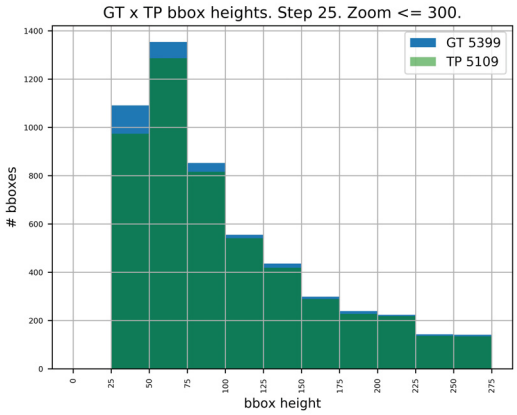
Figure 7.12 presents some of the executed experiments with minimum height restriction. It is possible to compare the ground truth bounding boxes by groups of increasing height to the successful predictions generated by the YOLOv3 model. The main conclusion when analyzing the charts is that the concentration of miss detections (visible slice of blue bars in the charts) is notoriously distributed below 100 pixels in height. However, removing such bounding-boxes did not provide meaningful improvements for the model's detection in the remaining heights.

Table 7.14: Four different configurations for the PTI01 dataset related to the minimum acceptable heights, evaluated by the YOLOv3 model. In each experiment, the column "Minimum Height" defines that every bounding box below the given height is removed from the training/evaluation, where the height "None" means any height and defines the baseline. The mAP metric is reported for both 40% and 50% of IoU. The "# bboxes" report how many bounding boxes are present in the dataset (train + test) for each configuration.

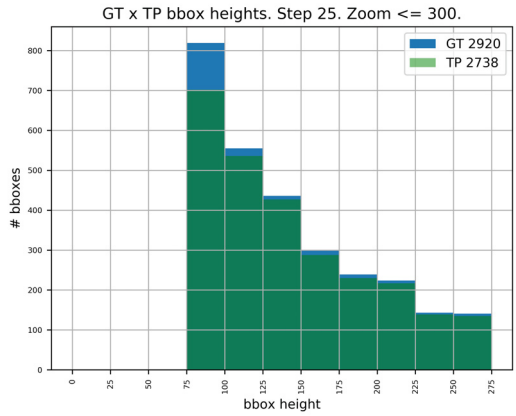
	Baseline	**Experiments**			
Minimum Height	None	25	50	75	100
mAP-40	0.9432	0.9433	0.9357	0.9344	0.9488
mAP-50	0.9332	0.9345	0.9287	0.9277	0.9451
# bboxes	30411	28202	22357	15460	11238



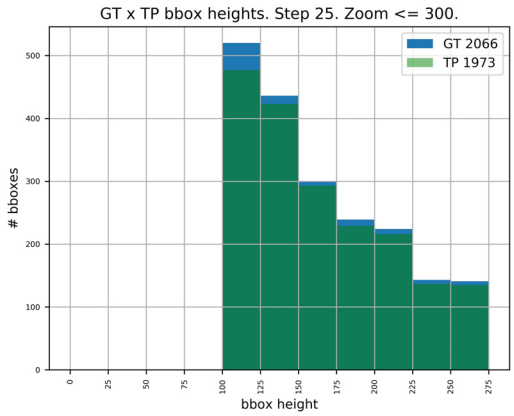
(a) Baseline: no height restriction



(b) Min-height 25 pixels



(c) Min-height 75 pixels



(d) Min-height 100 pixels

Figure 7.12: Default YOLOv3 tested in different versions of the PTI01 dataset which applied minimum height restrictions to the bounding-boxes. The baseline with no height restrictions is compared to other three versions of the dataset with a minimum height for the bounding boxes in 25, 75 and 100 pixels. The experiment with a minimum height of 50 pixels is not presented for brevity sake. The blue bars represent the ground truth bounding boxes while the green ones represent the True Positives predictions. The visible slice of each blue bar denotes the number of pedestrians that have not been detected by the model. The y-axis may have different scales in each chart.

7.1.7 Experimenting different mAP settings on PTI01

The mAP metric is the principal evaluation technique utilized in this work. As explained in Section 6.2, there are different settings for this metric which modify the detection quality exigency. The minimum IoU define this setting required to determinate a prediction as True Positive or not.

It is common in the literature (Everingham et al., 2010) to use the 50% of IoU in the mAP metric, which has been initially chosen as an arbitrary value. The COCO challenge (Lin et al., 2014) recently (Redmon and Farhadi, 2018) adopted the 75% IoU. In this experiment different IoU values are evaluated against the YOLOv3 and YOLOv3-tiny models in the PTI01 dataset with the "leaving events out" configuration. The results are expressed in Table 7.15.

Table 7.15: Comparison of different IoU values for the mAP metric in the PTI01 dataset with "leaving events out" setting on YOLOv3 and YOLOv3-tiny models.

IoU	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.75	0.8	0.9
	mAP									
YOLOv3	0.9263	0.9240	0.9198	0.9084	0.8793	0.7897	0.5893	0.5649	0.2546	0.0139
YOLOv3-tiny	0.8292	0.8242	0.8038	0.7672	0.7066	0.6098	0.4283	0.3075	0.1655	0.0071
	True Positives (TP)									
YOLOv3	4793	4782	4764	4716	4602	4267	3550	2961	2153	461
YOLOv3-tiny	4331	4311	4235	4100	3858	3461	2710	2165	1476	249

Percentual Performance Decrease comparing TP from lower and upper IoUs

IoU	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
YOLOv3	-	0.23%	0.38%	1.01%	2.42%	7.28%	16.80%	39.35%	78.59%
YOLOv3-tiny	-	0.46%	1.76%	3.19%	5.90%	10.29%	21.70%	45.54%	83.13%

According to the results, instead of using by default the 0.5 IoU metric value for the mAP, we choose to change the value to 0.4 slightly. In the YOLOv3 model, the difference of True Positives from 0.5 to 0.4 is of 114 detections, while for the YOLOv3-tiny the difference is of 242. Comparing the difference between 0.4 and 0.3 (48 and 135 respectively), the improvement is about half. Also, comparing 0.3 to 0.2 demonstrates a difference in True Positives of less than half. Such variation becomes more evident when looking at the "Percentage Performance Decrease" table's section. Analyzing the YOLOv3 model, from 0.1 to 0.2, the decrease in the number of TP is just 0.23%. From 0.2 to 0.3, this value increases to 0.38% (almost the double) but still is a very small quantity. The value reaches 1% according to the decrease presented in 0.3 and 0.4. From the 0.5 on, the performance decrease stats to rise significantly, always following the pattern of about doubling the previous value.

Because of this comparison, in our internal evaluations of PTI01, we change the default evaluation metric from mAP-50 (IoU equals to 50%) to mAP-40 (IoU equals to 40%) as a more meaningful setting, while keeping the variation ratio of performance decrease to a maximum of 1%. For the YOLOv3-tiny, we decide to keep the same metric configuration.

7.1.8 Caltech

In this section, the main experiments using the Caltech dataset are reported. Three dataset versions have been tested. The "Caltech10×" version, as the name infers, uses ten times more training data than the "Caltech1×". The "Caltech1× Sanitized" corresponds to the annotations that have been reviewed and improved by Zhang et al. (2016b). More details about those configurations can be verified in Section 6.4. Also, two metrics have been utilized: the mAP and the LAMR. The mAP is commonly used for evaluation of object detection tasks, and its precision parameter of IoU is usually defined as 50% (mAP-50), and for this evaluation, we have chosen to keep this value. The reported LAMR is considering the "reasonable" configuration which is the primary metric composition usually reported. More information about the metrics can be obtained in Section 6.2. Unless explicitly stated, the general training parameters are as follows: the data augmentation method is the "flipping only", the initial learning rate is set to 1^{-4}, and the anchors are set to the minimum number (3 for YOLOv3 and 2 for YOLOv3-tiny) which have been recalculated using our own k-mean implementation.

In Table 7.16, the YOLOv3 and YOLOv3-tiny are evaluated in three different configurations of the Caltech Pedestrian Dataset (Dollár et al., 2009b). At least, the following conclusions are obtained when analyzing the results. Training the YOLOv3 model with ten times more data enabled to achieve a lower LAMR (which is good) but also a lower mAP (which is bad). The YOLOv3-tiny has shown the same behavior for the Miss Rate metric but curiously did improve the mAP, which would be normally expected. We hypothesize that the YOLOv3 could be more sensitive to bad quality annotations, and even having ten times more data, it could not feed the model with relevant data.

Table 7.16: Main Caltech accuracy results for YOLOv3 and YOLOv3-tiny, reported on mAP-50 (the higher, the better) and LAMR on "Reasonable" metric (the lower, the better). The Caltech1× Sanitized version reported refers to the annotations provided by Zhang et al. (2016b). All the experiments are tested over Caltech1×, except the additional results marked with "*" which have been trained and tested on Caltech1× Sanitized.

Model	**Training Dataset**	**mAP-50**	**LAMR**
YOLOv3	Caltech10×	0.3465	0.1690
YOLOv3	Caltech1×	0.3511	0.2464
YOLOv3	Caltech1× Sanitized	0.3502 ; *0.5429	0.2019
YOLOv3-tiny	Caltech10×	0.2005	0.4298
YOLOv3-tiny	Caltech1×	0.1886	0.5270
YOLOv3-tiny	Caltech1× Sanitized	0.1929 ; *0.31	0.4595

Firstly, analyzing the models trained over both "1×" dataset, we see that the YOLOv3 model does not show a significant difference in accuracy when using the Sanitized version for the mAP-50 metric. Contrary to the YOLOv3 model, the YOLOv3-tiny does demonstrate to slightly decrease the accuracy due to the bad quality annotations. For both models, the LAMR show important differences in the score when using the Sanitized versions, reducing about 6.7% on tiny and 4.5% on the default.

For additional evaluation, we add the mAP-50 scores when both Sanitized experiments are also tested over a Sanitized test instead of the original ones. The quality in the new annotations enables the YOLOv3 model to increase by almost 20% in accuracy, while the YOLOv3-tiny improves in 12%. The results presented in this Section confirm that the YOLOv3 models show a higher potential concerning accuracy when fed with better quality data for both training and testing. We do not elaborate the same comparison using the LAMR metric due to unaffordable modifications that would be required in the Caltech's evaluation toolbox.

7.1.8.1 Trying different inference thresholds

The YOLO algorithm has a "score threshold" parameter which defines the minimum acceptable confidence for each proposed detection. The default score threshold is 0.3, that is, the detections are kept only if the confidence is above that value. Also, the YOLO works with an additional parameter called "IoU threshold" which is utilized by the Non-maximum Suppression (NMS) algorithm. The NMS is an algorithm that eliminates redundant predictions by filtering overlapping bounding boxes according to an IoU parameter, defined by default as 0.45. That is, for the predicted bounding boxes which have an intersection and have an IoU score higher than the "IoU threshold," all of them will be suppressed (removed) but the one with the highest confidence. In this section, we experiment changing those threshold parameters for the YOLOv3 model in the Caltech10× dataset to evaluate the performance variation in both mAP and LAMR metrics. In the remaining of this work, the "score" and "IoU" threshold values are kept in the default of 0.3 and 0.45 respectively, unless explicitly changed.

Table 7.17 presents the achieved results for the different "score threshold" values, with additionally showing the number of True Positives and False Positives for comparison. According to the results, the score threshold as 0.01 achieved the best performance in all metrics unless in the False Positives. It was possible to verify that decreasing the score threshold did increase the number of True Positives but at the cost of increasing the False Positives in four times when comparing 0.3 to 0.1 (10%) and, 42× when comparing 0.3 to 0.01 (1%). Despite the significant growth in the FP rate, mAP and LAMR metrics did not effectively reflect such variation. The mAP-50 increased by 8.83% and the LAMR decreased by 4.52%. For those metrics, elevating the TP value was much more meaningful than the increase in FP, mainly for the mAP metric which did improve by a higher value compared to the other metric.

Table 7.17: Comparison between different values for the "score threshold" value to generate predictions using the YOLOv3 model trained on the Caltech10× dataset. The mAP metric reports the IoU threshold in 50% (mAP-50) (the higher, the better), while the LAMR reports the best model in the "Reasonable" configuration (the lower, the better). Both True Positive (TP) and False Positives (FP) values are presented. The Ground Truth number of pedestrians is 4396. The best results are in bold.

Score threshold	mAP-50	LAMR	TP	FP
0.005	**0.4567**	**0.1285**	**3055**	27054
0.01	0.4558	0.1323	2925	17897
0.05	0.4312	0.1477	2425	4833
0.1	0.4109	0.1566	2201	2646
0.15	0.3952	0.1650	2060	1801
0.2	0.3821	0.1704	1958	1322
0.25	0.3572	0.1769	1862	1002
0.3 (default)	0.3684	0.1737	1727	640
0.4	0.3436	0.1866	1646	540
0.5	0.3323	0.1946	1582	433
0.6	0.3195	0.2119	1511	366
0.7	0.3017	0.2285	1415	**293**

Table 7.18 shows the achieved results when varying the "IoU threshold" while predicting. For this experiment, the "score threshold" was kept in 0.1. It is possible to see that for each metric there is a different optimal threshold. The mAP does better with 0.4, and the miss rate with 0.5. Both accuracy metrics perform with very similar scores when changing the threshold from 0.3 to 0.5. With a higher threshold, the True Positive rate did perform slightly better, probably because there are more bounding boxes per small areas in the image which enable the detection of pedestrians very close to each other, or even with occlusions. However, the increase in the threshold leads to higher False Positive rates once the NMS algorithm fail in suppressing redundant detections. By the other side, a very low threshold (0.05) reduced the number of False Positives, but also the True Positive rate.

We can conclude that if one is blindly seeking to improve the accuracy at all costs on any of both metrics, it is possible to sacrifice the False Positive rate to achieve better accuracy results. Also, to the best of our knowledge, the works which are in the top ranking of the Caltech Pedestrian Benchmark, such as SDS-RCNN (Brazil et al., 2017) and TLL-TFA (Song et al., 2018), do not report any information about the False Positive rates unless the "Miss Rate × FPPI" curve. Because of that, it is not clear whether they parameterize their detectors in such a way that it also generates high values of False Positives like we verified when using a score of 0.005 in the YOLOv3. In the next section, the "Miss Rate × FPPI" curve can be analyzed to provide a comparison between our detector with low score threshold to the works on the literature.

Table 7.18: Comparison of different values for the IoU threshold which is utilized as an overlapping criterion for the YOLOv3 predictions. The results are reported over the Caltech10× dataset using the mAP-50 and LAMR (Reasonable) metrics, and having the score threshold established in 0.1. The True Positive (TP) and False Positive (FP) values are presented. The Ground Truth number of pedestrians is 4396. The best results are in bold.

IoU threshold	mAP-50	Miss Rate x FPPI	TP	FP
0.05	0.393	0.1842	1835	**883**
0.1	0.401	0.1706	1856	1344
0.2	0.4069	0.1614	1911	1007
0.3	0.4103	0.1580	1902	1164
0.4	**0.4111**	0.1575	1933	1110
0.45 (default)	0.4109	0.1566	1939	1193
0.5	0.41	**0.1560**	1947	1325
0.6	0.4	0.1631	**1970**	1839
0.7	0.3726	0.1888	1825	2601
0.8	0.3352	0.2708	1902	5345

7.1.8.2 The best YOLOv3 model against the top tier methods on Caltech

In this section, we try to achieve the best possible score on the LAMR metric using the "Reasonable" configuration with the YOLOv3 model. The training is run on the Caltech10×. The parameters are modified according to previous experiments where they have been demonstrated to be promising. To get smoother training progress, we lower the initial learning rate from 1^{-4} to 1^{-5}, which will dramatically increase the number of required epochs but will also provide more stable accuracies between epochs (Section 7.1.1.1, Figure 7.4). We also activate the default data augmentation method provided by the YOLOv3 algorithm, which has demonstrated a slight increase in accuracy when using the complete model (YOLOv3) and the Caltech dataset (Section 7.1.5, Table 7.10). The anchors are set in three, one for each network's head, which has been recalculated specifically for the current dataset using our k-means implementation (Section 7.1.3, Table 7.4).

In Table 7.19, the YOLOv3 model is compared to the three methods that hold the state-of-the-art according to the accuracy rank in the official Caltech Pedestrian Detection benchmark. The "Caltech10×" is the dataset configuration chosen for comparison and the LAMR metric. The nine common metric configurations are reported (explained in Section 6.4), and three additional ones: "reasonable," "all aspect ratios" (AR All) and "large scale" (same as "Near Scale" but bounding boxes taller than 100 pixels).

The results reveal that the YOLOv3 (Redmon and Farhadi, 2018) model could not achieve the best scores in any of the compared metrics but in one. Both TLL-TFA and YOLOv3 achieved the maximum performance on the "Large Scale" metric configuration. The TLL-TFA (Song et al., 2018) has shown the best scores in 9 of the 12 evaluations. The SDS-RCNN (Brazil et al., 2017) demonstrated better results for the "reasonable" and the "partial occlusion" tests. The GDFL (Lin et al., 2018) has shown the best accuracy in the "typical aspect ratio" evaluation.

Despite the general bad results when compared to the top tier methods, the YOLOv3 did overcome some of them. The YOLOv3 did better than the SDS-RCNN in the "Overall metric" which evaluates the dataset "as it is," and also in the medium, large and far scale experiments, as well in the "Heavy Occlusion" and Near Scale. Despite not beating any of the compared methods on the "heavy occlusion" evaluation, YOLOv3 stayed only 3% behind of SDS-RCNN. It has also demonstrated better accuracy in the "Large Scale" and "Atypical Aspect Ratio" metrics compared to the GDFL method.

According to the main metric configuration, "Reasonable", YOLOv3 stays 5.49% behind the SDS-RCNN which holds the best score. The YOLOv3 does not show interesting scores in some metrics when compared to the TLL-TFA, like in the "Overall," "Heavy Occlusion" and "Medium Scale". In the remaining of the metrics, YOLOv3 overcome other methods like the SDS-RCNN or gets very close to them. YOLOv3 has demonstrated to be competitive in the majority of the metrics, being better than the SDS-RCNN in six configurations, and better than the GDFL in two. However, TLL-TFA demonstrates a solid difference in accuracy compared to YOLOv3.

Table 7.19: Comparing the YOLOv3 model to the three top methods reported in the Caltech's official benchmark. The methods are SDS-RCNN (Brazil et al., 2017), TLL-TFA (Song et al., 2018), GDFL (Lin et al., 2018) and YOLOv3 (Redmon and Farhadi, 2018). Twelve different configurations for the LAMR metric are reported according to the Caltech's benchmark tool. The YOLOv3 score and IoU inference thresholds are set as 0.005 and 0.45, respectively. The lower the metric value, the better the performance. The "AR" stands for Aspect Ratio and the "Occl." for occlusion. Values in bold are the best method in the metric. The green values highlight the metrics where the YOLOv3 wins over the orange marked methods. The table has been split into two parts for better display.

	Methods (1/2)			
Metric	SDS-RCNN	TLL-TFA	GDFL	YOLOv3
Reasonable	**7.36%**	7.40%	7.85%	12.85%
Overall	61.50%	**38.15%**	48.14%	53.11%
AR All	5.95%	**5.85%**	6.36%	11.47%
AR Atypical	11.68%	**9.09%**	14.67%	11.98%
AR Typical	4.58%	5.09%	**4.50%**	11.01%
No Occl.	5.95%	**5.85%**	6.36%	11.47%

	Methods (2/2)			
Metric	SDS-RCNN	TLL-TFA	GDFL	YOLOv3
Partial Occl.	**14.86%**	18.50%	16.74%	20.71%
Heavy Occl.	58.55%	**28.66%**	43.18%	49.56%
Near Scale	2.15%	**0.72%**	1.69%	1.54%
Medium Scale	50.88%	**22.92%**	32.50%	40.25%
Far Scale	100.00%	**60.09%**	70.97%	71.68%
Large Scale	0.97%	**0.00%**	1.14%	**0.00%**

Figure 7.13 presents the "Miss Rate × FPPI" curve using the YOLOv3 model with the best thresholds (score=0.005, IoU=0.5). The YOLOv3 demonstrates to have a worse curve, apparently due to the lower accuracy performance compared to the other works. However, we have not been able to define if the compared works have similar False Positive rate according to what has been verified in Section 7.1.8.1 when testing different thresholds for the YOLOv3 model. It is possible that the works in the top tier of the Caltech Benchmark have a very high number of False Positives, as YOLOv3 does in the current parameterization.

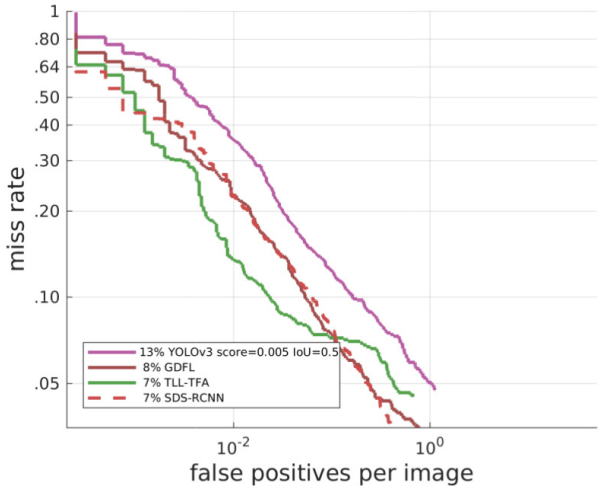


Figure 7.13: The "Miss Rate × FPPI" curve compares the YOLOv3 using the best threshold configurations (score=0.5%, IoU=50%) against the top 3 methods on the Caltech benchmark. The training follows the default Caltech10× configuration.

7.1.9 YOLOv3 versus YOLOv3-tiny

In this section, more information is reported about the comparison between the YOLOv3 and YOLOv3-tiny models. The evaluation is elaborated over the PTI01 dataset using the "leaving events out" setting, and also with the Caltech dataset using the "10×" configuration. The results are reported using the mAP metric. Observe that the dimensions (axis) may vary in some charts according to the slice of data utilized for better presentation, or to the dataset utilized.

Figure 7.14 presents the True Positive detection against the Ground Truth bounding boxes distributed in 50 groups of box areas which are below to 2,000 (pixels). It is possible to see that both models tend to decrease their performance according to smaller bounding boxes areas, while YOLOv3-tiny demonstrates a significant increase in this behavior. The Caltech dataset presents a higher quantity of small-area bounding boxes, which are very concentrated around the 500 pixels. The PTI01 has a concentration of bounding boxes in the range of 250 pixels, having more distributed quantities according to the increase in the size of the areas. The overall performance of YOLOv3 on Caltech is way worse when compared to the PTI01. It is important to denote that small bounding boxes are not necessarily related to small-scaled pedestrians, once it can refer to people's body parts, head, or occluded pedestrians.

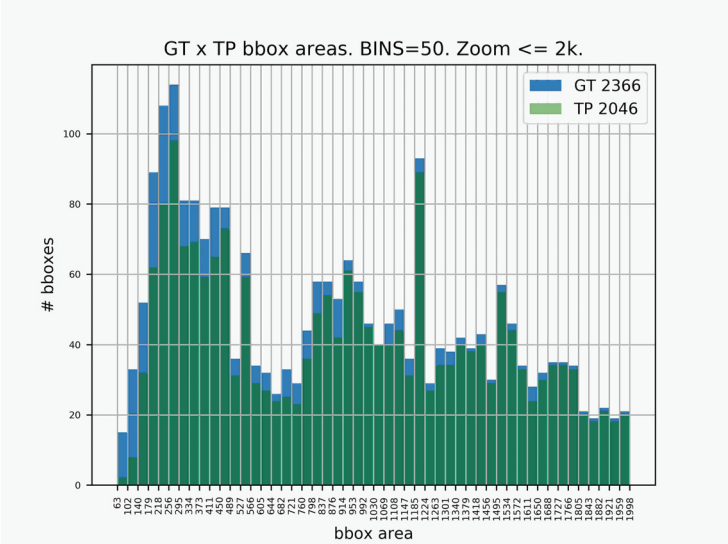
In Figure 7.15, both models are compared using the bounding boxes ratios for the True Positives and Ground Truth. In the PTI01 dataset, both models perform worse in the detection of bounding boxes below the 3.0 ratio when compared to the ones above the value. For the Caltech, the referential ratio is about 2.3, with a much lower general performance. This effect is better observed in the YOLOv3-tiny evaluation which demonstrates a poorer detection quality.

A comparison using bounding box heights is shown in Figure 7.16. As expected due to the results provided by the bounding box area evaluation (Figure 7.14), both models produce lower quality detections for smaller heights, while they perform similarly to each other in heights above ~160 pixels for the PTI01 dataset and 100 pixels for the Caltech dataset. However, the Caltech dataset presents a high concentration of bounding boxes for heights below~75 pixels, while PTI01 has a smoother distribution within higher heights.

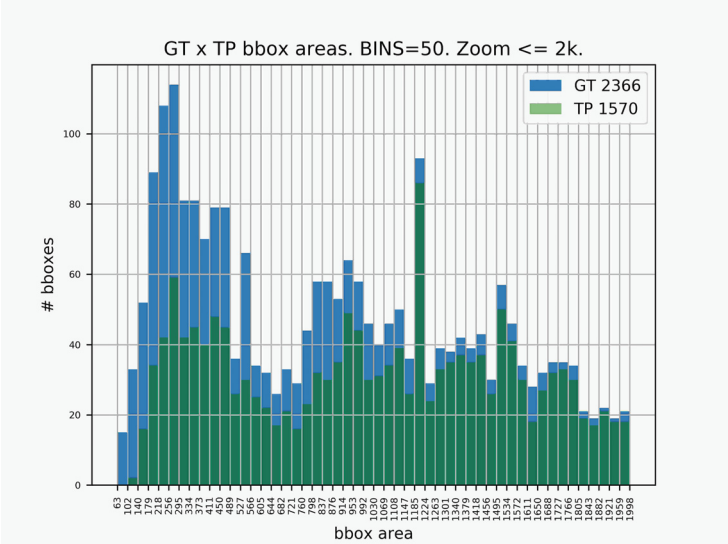
Figure 7.17 uses the Miss Rate × FPPI curve to compare the performance of both models in the PTI01 and Caltech datasets. In both scenarios, the default model (YOLOv3) shows the achievement of lower Miss Rates at the smallest amounts of FPPI. The ROC curve in Figure 7.18 also demonstrates the higher performance of the YOLOv3 model in both datasets, and the higher difficulty faced in the Caltech. In the last curve, YOLOv3-tiny presents a higher False Positive rate denoting that the model produces more than the double of FPs in order to achieve high TP.

In the same context, in Figure 7.19 it is possible to verify more clearly that the YOLOv3 model achieves greater True Positive quantities and lower False Positives compared to the YOLOv3-tiny. In the Caltech dataset, the YOLOv3 model generates a higher proportion of FPs compared to the PTI01 dataset. In the PTI01 the proportion is ~5% of FPs while in the Caltech it reaches ~37%. For the YOLOv3-tiny the values are respectively ~16% and ~81%. This demonstrates that the models not only struggle to generate TPs but also produces many FPs in the process. This behavior is also reported in the ROC curves (Figure 7.18)

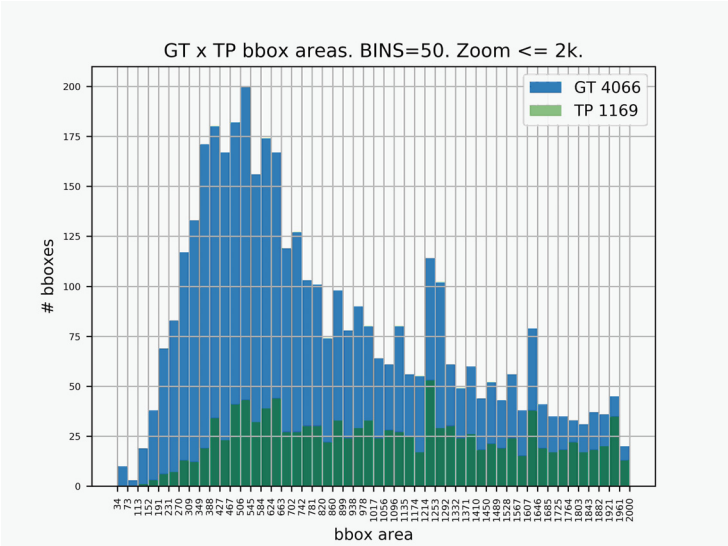
Both models (default and "tiny") have been trained in the PTI01 dataset with a single class. In order to understand in which appearance categories each model did perform better, the "detranslation" method is utilized (explained in Section 7.1.4). Every ground truth bounding boxes, all belonging to the same "person" class, are linked back to the eight original classes. Surprisingly, the YOLOv3-tiny did not perform so worse to every class. It has shown similar performance for the classes "angled," "no occlusion," "body parts," "head" and "top view." According to previous experiments, the latter three classes are known to being the hardest classes in the PTI01 dataset. The YOLOv3 demonstrates a better detection quality for the classes "far,"



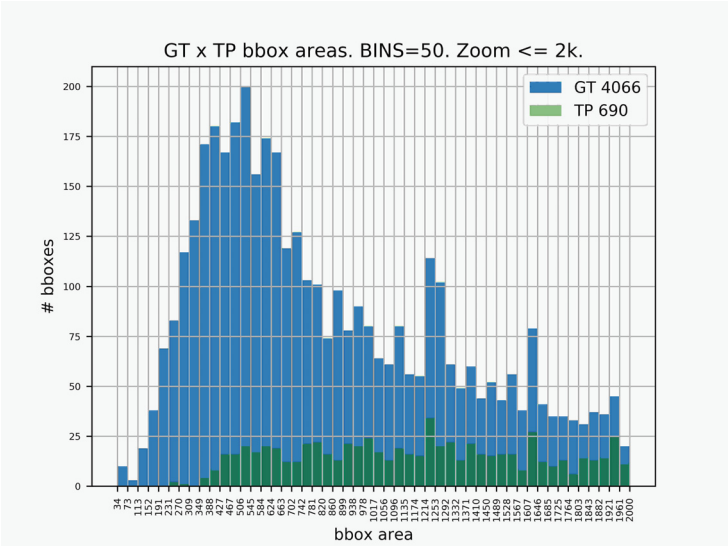
(a) YOLOv3



(b) YOLOv3-tiny



(c) Caltech10×: YOLOv3

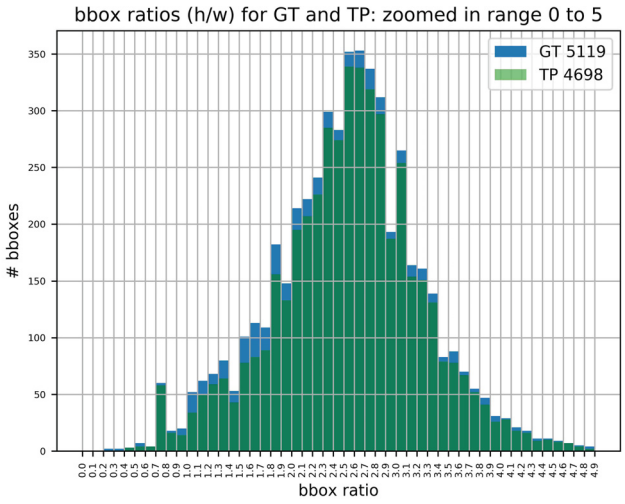


(d) Caltech10×: YOLOv3-tiny

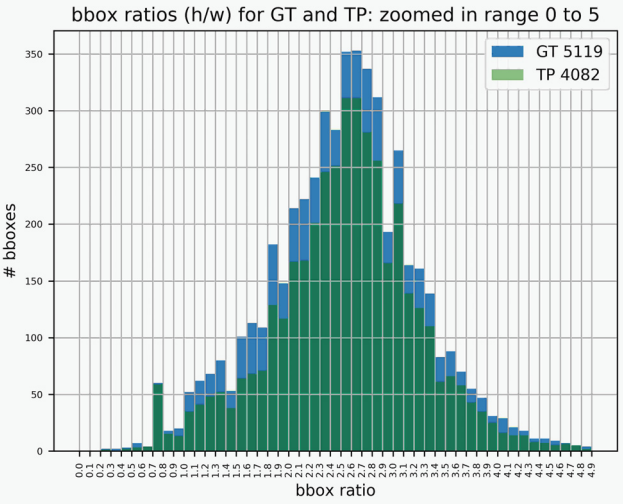
Figure 7.14: Comparison between pedestrian detection by bounding box areas in the default YOLOv3 and YOLOv3-tiny on PTI01 "leaving events out" setting, and on Caltech with the "10×" configuration. Both models demonstrate lower performance for smaller bounding boxes areas, wherein YOLOv3-tiny accentuates such behavior. Ground Truths in blue and True Positives in Green.

"normal occlusion" and "severe occlusion." This proves that the YOLOv3-tiny does not perform so well for small-scaled or occluded pedestrians as YOLOv3 does. The situation changes when considering high-biased datasets, such the one experimented in Section 7.1.4 which YOLOv3-tiny did perform well for the "far" class. This evaluation could not be executed in the Caltech dataset due to the non-existence of such extra classes.

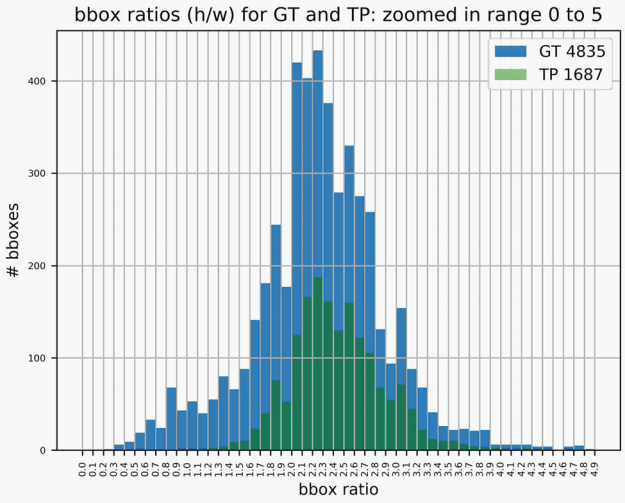
In terms of accuracy, as a general conclusion, YOLOv3-tiny demonstrated to achieve lower performance compared to the default model, but still interesting close. In both datasets, the difference in accuracy (mAP) for both models is about 14%. The main gap in the YOLOv3-tiny model, demonstrated to be related to small-scaled and occluded pedestrians. Figure 7.24 (in the end of the chapter) presents some samples of YOLOv3 and YOLOv3-tiny predictions on the PTI01 dataset. Despite the general worse performance of YOLOv3-tiny, in the figure it is possible to see that sometimes the YOLOv3 simple do not detect clear and close pedestrians



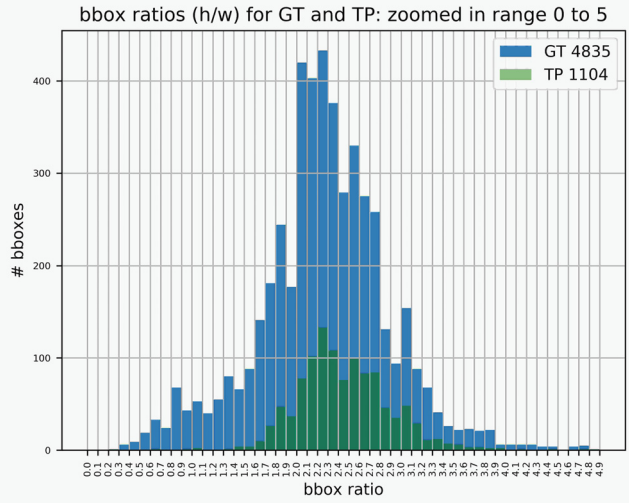
(a) PTI01: YOLOv3



(b) PTI01: YOLOv3-tiny



(c) Caltech10×: YOLOv3



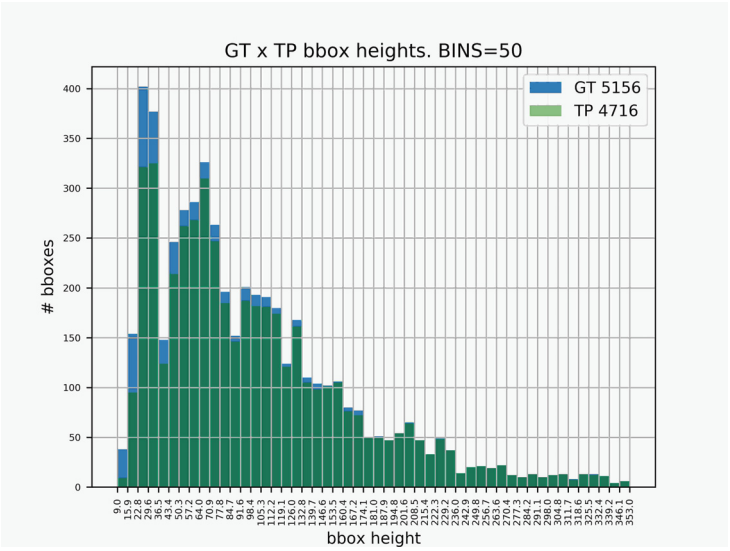
(d) Caltech10×: YOLOv3-tiny

Figure 7.15: Comparison between pedestrian detection by bounding box ratios (height/width) in the default YOLOv3 and YOLOv3-tiny on PTI01 "leaving events out" setting, and on Caltech with the "10×" configuration. YOLOv3-tiny demonstrates lower performance on general bounding boxes ratios. The models show quite similar proportional performance for the different datasets, but the Caltech proves to be harder. Ground Truths in blue and True Positives in Green.

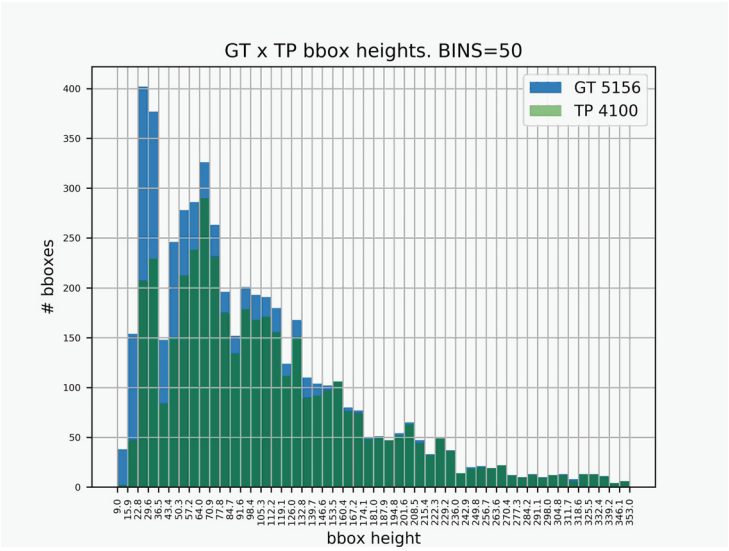
(top-right) or simple make predictions where there is no pedestrians (fifth and last rows on left, and third row on the right).

7.1.10 YOLOv3 training and inference timings

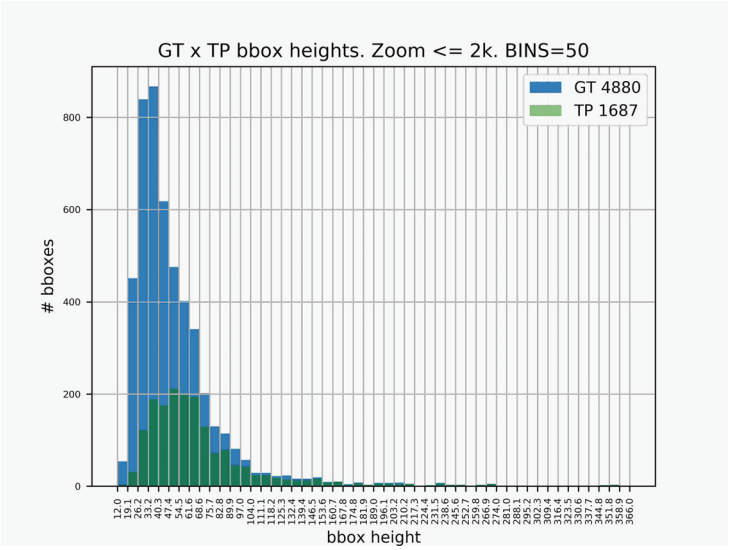
Table 7.20 presents the average inference speed in Frames per Second (FPS) for the YOLOv3 and YOLOv3-tiny models, while Table 7.21 reports the approximated number of hours required to train the models on Caltech and PTI01 datasets. In both datasets, the images are in the 640×480 pixels of resolution, and the models have been trained and tested on an NVIDIA TITAN XP and a TITAN X GPUs.



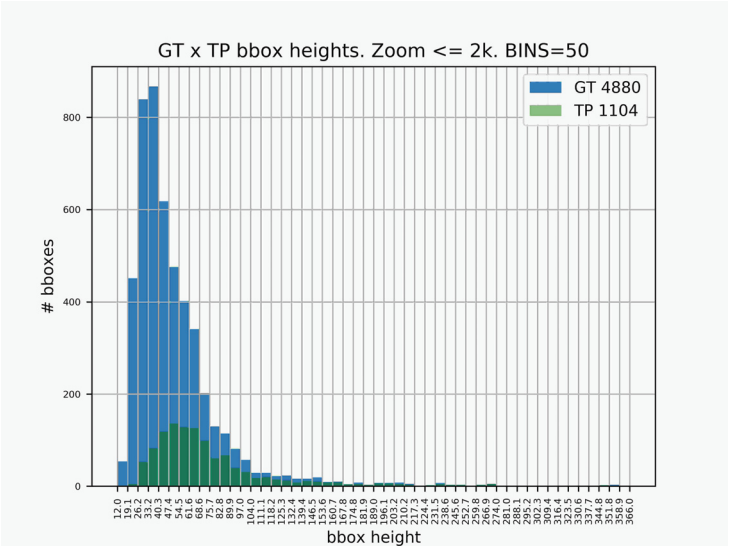
(a) PTI01: YOLOv3



(b) PTI01: YOLOv3-tiny



(c) Caltech10×: YOLOv3



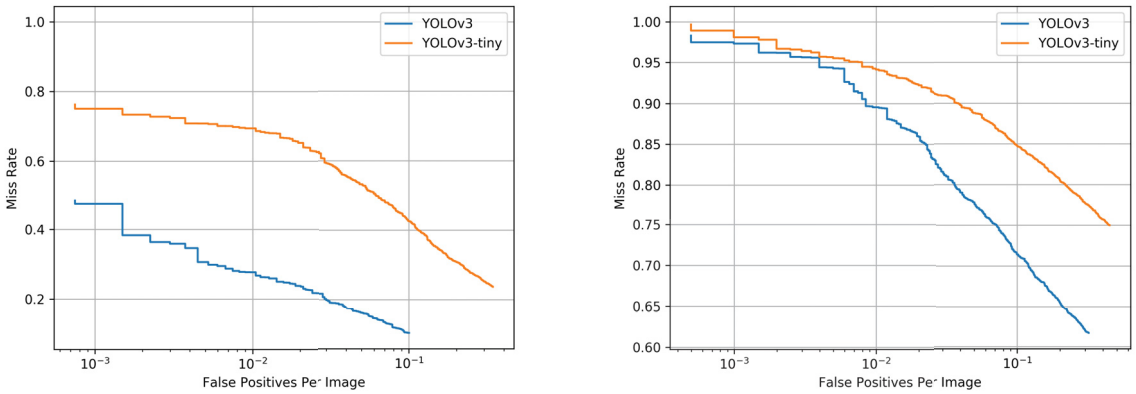
(d) Caltech10×: YOLOv3-tiny

Figure 7.16: Comparison between pedestrian detection by bounding box heights in the default YOLOv3 and YOLOv3-tiny on PTI01 "leaving events out" setting, and on Caltech with the "10×" configuration. The YOLOv3-tiny model potentiates the natural lower performance of the YOLOv3 model for smaller bounding boxes. Ground Truths in blue and True Positives in Green.

Table 7.20: Inference speed in Frames per Second (FPS) for YOLOv3 and YOLOv3-tiny for images with dimensions of 640×480 pixels which corresponds to the PTI01 and Caltech datasets. The input of the network is also 640×480. Inferences are executed in the Darknet framework, as well in the Keras implementation of YOLOv3, using both NVIDIA Titan X and XP GPUs.

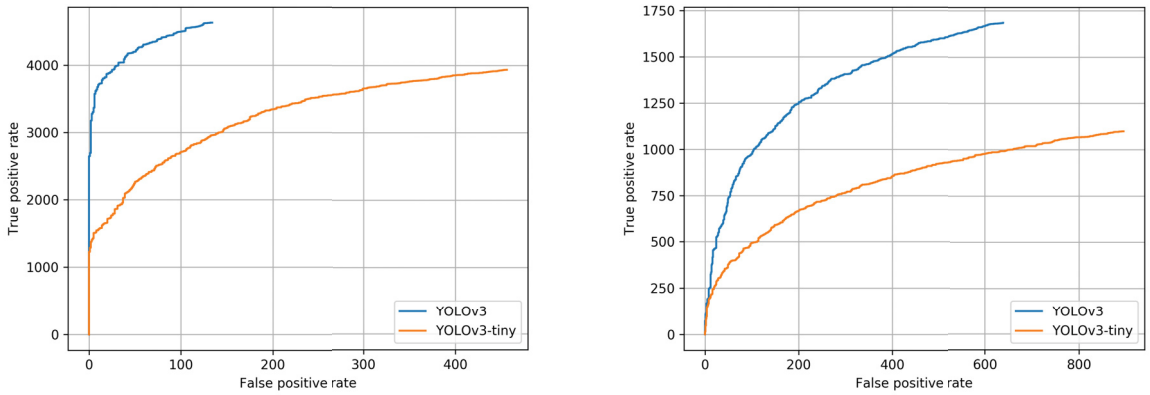
		FPS			
	GPU	TITAN XP		TITAN X	
	Framework	Keras	Darknet	Keras	Darknet
Model	YOLOv3	19	34	13	27
	YOLOv3-tiny	43	169	25	121

The Keras implementation demonstrated to be slower than the original one implemented in the Darknet framework. When comparing both frameworks speeds, for the YOLOv3 model,



(a) PTI01 (b) Caltech10×

Figure 7.17: Comparison between Miss Rate × FPPI curves for the default YOLOv3 and YOLOv3-tiny on PTI01 "leaving events out" setting, and on Caltech with the "10×" configuration without any bounding box filtering.



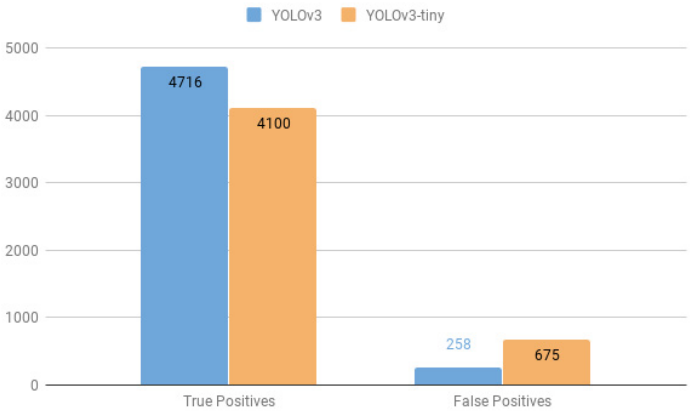
(a) PTI01 (b) Caltech10×

Figure 7.18: Comparison between ROC curves with True Postive and False Positive rates for the default YOLOv3 and YOLOv3-tiny on PTI01 "leaving events out" setting, and on Caltech with the "10×" configuration. YOLOv3 demonstrates a better curve while achieving less than half of the False Positive rate.

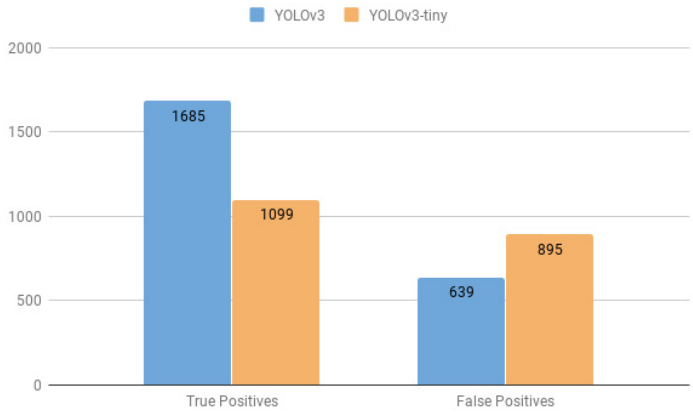
Table 7.21: Training speed of the YOLOv3 and YOLOv3-tiny models on the datasets: PTI01 with "leaving events out", Caltech10× (Dollár et al., 2009b) and Caltech1× Sanitized (Zhang et al., 2016b). The speed is defined by the amount of time, in hours, required to completely train a model using two different data augmentation methods. The models have been trained using a YOLOv3 implementation of Keras + TensorFlow executed in an NVIDIA Titan XP GPU.

	Training hours			
Model	YOLOv3		YOLOv3-tiny	
Data Augmentation	Flip	Default	Flip	Default
PTI01 (leaving events out)	3.8	9.75	0.71	8.52
Caltech10×	45	58.2	7	43.5
Caltech1× Sanitized	1.95	5.2	0.33	2.85

the Keras has about half of the speed of the Darknet, while for the YOLOv3-tiny the Darknet shows to be four times faster. When comparing the speeds of the models, it is possible to verify

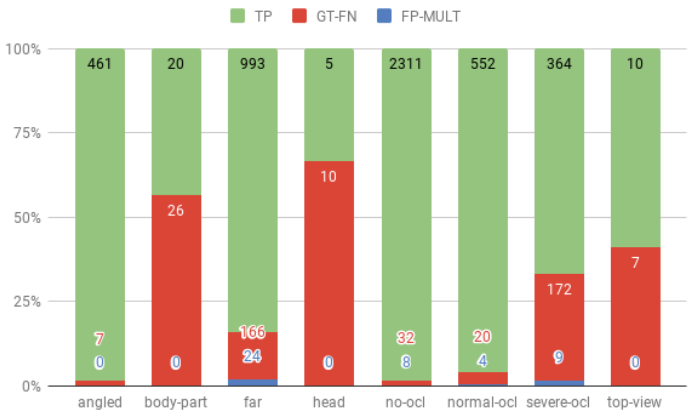


(a) PTI01. Ground Truth number: 5156.

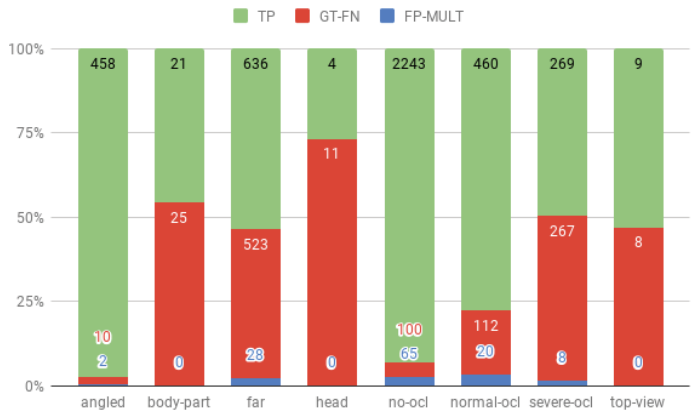


(b) Caltech10×. Ground Truth number: 4396.

Figure 7.19: Comparison between the quantities of True Positive and False Positive predictions for the default YOLOv3 and YOLOv3-tiny on PTI01 "leaving events out" setting and on Caltech with "10×" setting. YOLOv3-tiny demonstrates a lower number of TP and a greater quantity of FP on both datasets.



(a) PTI01: YOLOv3



(b) PTI01: YOLOv3-tiny

Figure 7.20: Comparison between the quantities of True Positives (TP in green), False Negatives (GT-FN in red) and False Positives by Multiple Detection (FP-MULT in blue) predictions for the default YOLOv3 and YOLOv3-tiny for each original class of PTI01 dataset in the "leaving events out" setting. Both models have been trained with a single class (person) wherein the predictions have been linked with the original 8 class from the dataset. YOLOv3-tiny shows similar performance to YOLOv3 default but for classes "far," "normal-ocl" and "severe-ocl."

that the YOLOv3-tiny is two times faster than the YOLOv3 in the Keras framework, and five times faster when using the Darknet.

Observing Table 7.21, the amount of time required to train the models on Caltech10× demonstrates to be considerably higher than the other two dataset configurations. To train YOLOv3 on Caltech10×, it takes two and a half days using the default data augmentation method. The Caltech1× Sanitized provides the fastest training for any of the evaluated model configurations. According to the training hours required, it is possible to verify that the data augmentation methods do impact the amount of time needed to train a model. Analyzing the Caltech10×, YOLOv3 takes 13 hours more to finish the training when using the default data augmentation model provided by the YOLOv3 algorithm, instead of the "flipping only" method.

Interestingly, for that same dataset the YOLOv3-tiny takes only 7 hours to complete the training but dramatically increases the required time to 43.5 hours (an increase of 36 hours).

This behavior may be related to the higher complexity in the pedestrian appearance generated by the data augmentation method which may overcome the knowledge extraction capabilities available in such a smaller version of the YOLOv3 model. This situation has also been identified in Section 7.1.5, where the YOLOv3-tiny achieves greater accuracy when using the "flipping only" technique with a much lower number of epochs compared to the default method.

7.1.11 General conclusions about YOLOv3

In Section 7.1 different configurations have been tested for the YOLOv3 and YOLOv3-tiny models using the datasets PTI01 and Caltech. Even the datasets have been evaluated in different settings. The main idea was to explore the fundamental behavior of such models in the proposed scenarios. Also, the comparison with other methods has been developed. In this section, the overview of the main results is discussed.

In the first analysis (Section 7.1.1), it was possible to verify the impact of lower and higher quality annotations when training the models. For both PTI01 and Caltech datasets, when using grounding truth annotations with the better quality, the models did make use of more epochs before reaching the "early stopping" configuration. That is, the model could get the "validation loss" decreasing for more iterations, while when using the lower quality annotations the models did abort the training a few iterations earlier. We raise the hypothesis that the model could work with useful information for a longer time. In other words, supposedly, the bad quality annotations did disturb the training process by providing uncommon or unexpected patterns in a higher frequency than the sanitized annotations. As expected, the detection quality is directly related to the quality of the training data. Additionally, Figure 7.3 may infer that the YOLOv3 is able to naturally suppress some of the bad quality annotations provided in the Caltech1× original set.

In the same section (7.1.1), we see that decreasing the validation loss does not necessarily translate to better final accuracy. Despite the loss keeping to decrease along the iterations, when testing, the epochs with better accuracy are not usually the last ones. Such variations may be related to the effect of over-training. The model successfully keeps decreasing the loss according to the validation data, but when checking the final test data, the accuracy oscillates.

In order to experiment on a similar idea to the one proposed by Dollár et al. (2009b), where they have standardized the pedestrian bounding boxes to a fixed ratio, we have proposed to evaluate the effect of defining multiple fixed ratios which have been equally distributed into the major range of common ratios in the PTI01 dataset. With this procedure, we have created, called by us, the "canonical bounding boxes." Many combinations of the newly generated bounding boxes have been tested, using them for training, testing and even standardizing them in a post-processing manner (Table 7.1). However, this technique demonstrated to only decrease the model's performance, for both YOLOv3 and YOLOv3-tiny. If the YOLO benefits from such bounding box modification, it may be in a different way from the one applied in those experiments.

Curiously, the default recalculation procedure for the anchor boxes did not generate the expected effect in the related experiments (Section 7.1.3). Instead of improving the model's accuracy, the default anchor recalculation algorithm provided a dramatic in the model's performance. We have found that using a simple clustering algorithm implementation, such as k-means, the results were better than using the default anchors and much better than using the default algorithm. Additionally, it was possible to verify that it was not worth it of using more than one anchor per model's head. For the YOLOv3, using three anchors provided better results than using the default nine. As well, the YOLOv3-tiny demonstrated higher accuracy with two anchors rather than six.

Multiple classes have been introduced to the PTI01 dataset to enable the understanding of the performance of the YOLOv3 model according to variations of the pedestrian appearances about the different camera views. Each class represents pedestrians which have been grouped due to the similarity of their views on the cameras. For instance, they have been grouped in people distant from the camera, people that are recorded from a top view, pedestrians which are occluded by objects, and other situations which change the default signature of the aspect of a pedestrian. Due to this experimentation, it was possible to find that the model better handles the pedestrians which are far from the camera compared to the ones which are occluded (Figure 7.8(b)). Among the eight classes, this one with people in a far view is the third easiest kind of pedestrian to be detected by the YOLOv3, losing only to person with the standard view (no occlusions) and person which are angled in the scene (corners of the camera). Another important confirmation is that the most challenging pedestrians to be detected are the ones from a top view angle, and body parts (including head). This may be related to the high variation of the typical structural appearance of a person when, for instance, viewed from above compared to the other views (Figure 5.3).

To promote further evaluation of the influence of appearances variations in the model's performance, different configurations of these classes have been tested while training. Many combinations of merging different classes and removing others from the dataset have been elaborated in order to understand the impact of those when not present in the dataset or when combining pedestrians in different views. The results demonstrated that the YOLOv3 model, in general, works better when using a single class to represent all the pedestrians in the dataset, rather than any other combination. In truth, removing the hardest classes from the dataset did slightly improve the model's performance. However, removing those classes did not significantly improve the performance of the other classes. In such a way, it is worth to keep such hard classes letting the model detect at least part of the pedestrians than detecting no one. The conclusion is that the class division was relevant to understand the performance of the models according to the pedestrian appearance, but for practical detection training, a single class is enough.

Three different data augmentation configurations have been tested in both PTI01 and Caltech datasets (Section 7.1.5). The YOLOv3-tiny demonstrated to benefit more from the "flipping only" technique (which randomly flips the images horizontally) compared to the default and more elaborated augmentation method. Hypothetically, this behavior may be related to the lower complexity of the "tiny" network which could find the variations of the pedestrian appearance too difficult when using the default method. The YOLOv3 has shown slightly better performance when using the "flipping only" on the PTI01 dataset while benefiting more from the default method when testing on Caltech. These results may be related to the fact that, in general, the PTI01 dataset demonstrated to be a much easier dataset compared to the Caltech. It has higher quality annotations in a more stable scenario. The Caltech contains many bad annotations, and the scenes are constantly changing because the dataset was built from a camera in a moving car. We theorize that the default data augmentation method introduces to the YOLOv3 model a higher level of appearance variation which is closer to the Caltech's situation, but an almost too strong variation when looking to the PTI01 dataset. Additionally, it has been verified that the default data augmentation method decreases the model's performance for the detection of occluded pedestrians (Table 7.10). It seems like the more substantial variation in the appearance disturbs the model's understanding of the occluded pedestrians, but to this topic, it would be required more study to identify the exact relation between the lower occlusion performance when using this method.

The PTI01 dataset has been classified in different levels of difficulty according to the variations in the configurations of the training and testing sets (Section 7.1.6). Observing the

performances of YOLOv3 and YOLOv3-tiny on those configurations, the "leaving events out" has demonstrated to be the intermediate and feasible difficulty. Also, the YOLOv3 model has shown an impressive ability extracting knowledge with smaller quantities of training data. Reducing the data to $\frac{1}{3}$, YOLOv3 did reduce less than 3% in the mAP metric.

When comparing the YOLOv3's performance to other top-tier methods in the Caltech dataset (Table 7.19), YOLO stays considerably behind when looking to the "Reasonable" metric setting. It has almost the double (1.7×) of the "Miss Rate x FPPI" compared to the best method: the SDS-RCNN (Brazil et al., 2017). However, interestingly, YOLOv3 surpass the SDS-RCNN in 6 other settings, including "Heavy Occlusion," "Far Scale" and "Overall". Unfortunately, it does not mean the YOLOv3 achieved the top performances for those settings once they have been achieved by the TLL-TFA (Song et al., 2018) method. Nevertheless, both TLL-TFA and YOLOv3 show the maximum performance in the "Large Scale" setting, that is, 0% of LAMR. The TLL-TFA rules the majority of the evaluated settings (10 of 12) demonstrating to have a more solid pedestrian detection compared to other methods.

Despite not being so accurate as the top works on the Caltech's benchmark, YOLOv3 has demonstrated to be the fastest among the compared methods (Table 7.22). Curiously, the TLL-TFA (Song et al., 2018) does not report their inference speed, and thus it is not possible to compare it. Reasoning about this fact, hypothetically, the motivation of not reporting the detector's speed could be related to a non-competitive performance. The GDFL method (Lin et al., 2018) demonstrates to be faster than the other methods but still behind YOLOv3 (7 FPS below YOLOv3). However, it holds top-tier accuracy in the Caltech challenge, achieving less than half of the LAMR compared to YOLOv3.

Table 7.22: Inference speed in Frames per Second (FPS) for the compared methods: SDS-RCNN (Brazil et al., 2017), GDFL (Lin et al., 2018), TLL-TFA (Song et al., 2018). The missing information which has not been given by the authors are marked with a "-".

Method	**FPS**	**Framework**	**GPU**
SDS-RCNN	5	Caffe + Matlab	TITAN X
GDFL	20	-	GeForce GTX 1080 Ti
TLL-TFA	-	-	-
YOLOv3	27	Darknet	TITAN X
YOLOv3-tiny	121	Darknet	TITAN X

7.1.11.1 Accuracy versus speed tradeoff

In order to help in the reasoning about the relation between accuracy and speed in practical terms for a pedestrian detector, a hypothetical scenario is elaborated using real information from the PTI's video surveillance network and the performances of the evaluated detectors. The idea is to simulate the ability to detect pedestrians using the referred detectors in a real-world scenario.

Firstly, the scenario is conjectured. Using the PTI's camera network, it is possible to calculate an estimate of 10.7 millions of frames being daily recorded by more than 250 cameras (Table 5.1). For calculation purposes, arbitrarily, it is defined that there are three people always appearing in each frame. If that number were real, about 32.2 million pedestrians would appear in the PTI's video surveillance network in a single day. That means a perfect pedestrian detector should correctly find all of those people and only those people. Changing the number of people per frame would escalate the calculation, but the proportion in the reasoning would be the same.

In a second step, the detection performances of the four models are compared. The calculations are summarized in Table 7.23. Initially, the speed of inference of each model is translated to the maximum number of frames it could detect using a single instance on similar GPUs. As well, for the imaginary number of pedestrians per frame being 3, the respective maximum number of detection per day is calculated. The TLL-TFA (Song et al., 2018) is unconsidered once there is no information about the inference speed which is a mandatory parameter in this analysis.

Table 7.23: Calculations on a simulated environment based on the PTI's video surveillance network, to evaluate the estimate the detection efficiency of the compared methods. The values are based on a single day of video recording from the PTI's camera network. The inference speed and accuracies are also reported.

Parameter	**Value**			
Frames per day	10,749,897.00			
Pedestrians per frame	3			
Pedestrians per day	32,249,691.00			
CALCULATIONS				
Method	SDS-RCNN	GDFL	YOLOv3	YOLOv3-tiny
GPU	TITAN X	GeForce GTX 1080 Ti	TITAN X	TITAN X
FPS	5	20	27	121
Recall	70.00%	70.00%	70.00%	70.00%
Precision	97.00%	97.00%	92.00%	75.00%
Frames processed per day	432,000.00	1,728,000.00	2,332,800.00	10,454,400.00
Maximum possible detections	1,296,000.00	5,184,000.00	6,998,400.00	31,363,200.00
Sucessfull detections (True Positives)	907,200.00	3,628,800.00	4,898,880.00	21,954,240.00
Missed pedestrians (False Negatives)	388,800.00	1,555,200.00	2,099,520.00	9,408,960.00
False Positives	28,057.73	112,230.93	425,989.57	7,318,080.00
Total number of predictions	935,257.73	3,741,030.93	5,324,869.57	29,272,320.00
Number of instances to real-time	24.88	6.22	4.61	1.03
% of processed frames compared to PTI's total	4.02%	16.07%	21.70%	97.25%

The Precision × Recall curves generated by the Caltech's benchmark contrast the accuracy performance of the SDS-RCNN (Brazil et al., 2017), TLL-TFA (Song et al., 2018), GDFL (Lin et al., 2018), YOLOv3 and YOLOv3-tiny (Redmon and Farhadi, 2018) (Figure 7.21). Recall ratio at 70% supports the remaining calculations. Using that Recall, an approximated value of Precision is collected based on the curves. Now that the maximum values in frames, pedestrians, and scores for Recall and Precision are established, it is possible to estimate the number of correctly detected pedestrians, missed pedestrians and false positives.

According to the calculated values, some conclusions can be inferred. The SDS-RCNN, being the most accurate detector in the Reasonable setting, if using a single instance processing frames for 24 hours it would be able to handle only 4% of the frames generated by the PTI's camera network in a single day. That means that for processing the frames in real-time, it would be necessary 25 instances with 25 dedicate GPUs. In another point of view, if a single instance is kept, the SDS-RCNN would take 25 days to process a single day of the recorded data. The GDFL would be able to process 16% of the data. A single instance of the YOLOv3 would process about 21% of the same data, requiring only five instances to handle it in real-time. This translates in a saving of 5 times in terms of resources. The YOLOv3-tiny is so fast that could handle the entire data em real-time using almost a single instance.

Despite being the faster, YOLOv3-tiny presents the highest False Positive rate. If there would be three pedestrians per frame, the YOLOv3-tiny are going to process 43 million in a single day, where 10.2 million are False Positives, and 30.6 are correctly detected pedestrians. The SDS-RCNN does generate only 26.7 thousands of false positives, but at the same time, it is going to detect only 864 thousand real pedestrians. The GDFL shows a closer performance compared to the YOLOv3, detecting 3.6 millions of pedestrians while having 112.2 thousands of False Positives. On the other hand, the YOLOv3 is going to detect 6.1 millions of pedestrians

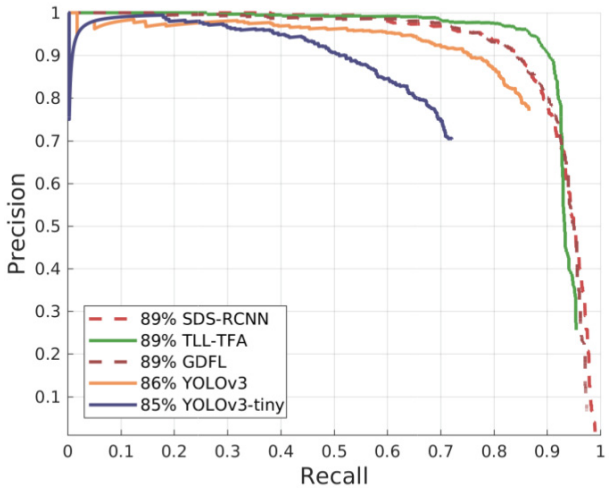


Figure 7.21: The Precision × Recall curve compares YOLOv3 and YOLOv3-tiny with the top methods in the Caltech Pedestrian Detection benchmark: SDS-RCNN (Brazil et al., 2017), GDFL (Lin et al., 2018) and TLL-TFA (Song et al., 2018).

(True Positives) and 536.4 thousands of false predictions. That is, YOLOv3 detects 35% more pedestrians than GDFL and 3.8 times more False Positives. Compared to the SDS-RCNN, the YOLOv3-tiny detects 5.4 times (540%) of the number of pedestrians detected by GDFL and more than 15 times the number of false pedestrians.

This analysis demonstrates that YOLOv3 and YOLOv3-tiny would be able to find a much higher number of pedestrians in the video surveillance scenario of the PTI, in a same period, compared to the other methods. Nevertheless, the number of false detection dramatically increases, and depending on the application this would prejudice the task. The pedestrian detection task does not operate alone. For example, in the Person Re-identification task, the pedestrian detection is just the first of three sub-tasks. The tracking and the identification, usually are subsequent tasks which use the pedestrian detector predictions as inputs. If the other two tasks are faster as the detector is, the number of False Positives may not cause significant processing overhead. However, if the remaining of the pipeline is much slower, the pedestrian detection speed does not translate to general efficiency. In such case, any additional false positive to be processed would be very expensive when seeking for maximum efficiency. In this reasoning, the YOLOv3 and YOLOv3-tiny models demonstrated to be fast and produce an impressive number of correct detections, but their applications may be weighted by the impact of the higher number of False Positives.

7.2 WEAK SEMANTIC SEGMENTATION IN YOLOV3

In this Section, the main results are related to the YOLOv3 experimentation with the “weak semantic segmentation” infusion are reported. The infusion technique is based on the SDS-RCNN (Brazil et al., 2017) implementation which demonstrated improvements in the Pedestrian Detection accuracy without affecting the inference speed of the model. In order to verify whether such a technique has a positive effect on YOLOv3, a similar implementation is developed and tested. More information about the proposal of this experimentation is described in Section 5.3.

7.2.1 Selecting the configurations for the experiment

Two versions of the model are compared to evaluate the effectiveness of the infusion technique when using the YOLOv3 to detect pedestrians. The first is the YOLOv3 model in its original format. The second is the YOLOv3 model in which the original neural network is modified to include a segmentation "head" according to the implementation described in Section 5.3.

The "T-test" statistical test is utilized to compare multiple instances of both versions of the model (Section 6.6), providing a precise judgment about the infusion effectiveness. About ten instances of each model version, in each configuration, is utilized for the comparison.

Both YOLOv3 and YOLOv3-tiny models are tested, wherein they have specific infusion implementations according to the different network structures. Also, both Caltech and PTI01 datasets are utilized in the experimentation. Two versions of each dataset have been chosen to increase the variability of the scenarios where the infusion technique are tested. For the Caltech dataset the two configurations are: the "10×" (Dollár et al., 2009b) as the setting utilized by Brazil et al. (2017) when training the SDS-RCNN model, and the "1× Sanitized" provided by Zhang et al. (2016b) which is selected as a possible easier configuration. For the PTI01, also two configurations are selected: the first is the "leaving events out" as a default challenge and also the "leaving 3 cameras out & discarding" which represents a harsh setting where the model is tested in 3 cameras never seem in training and using only 33% of the training data.

Additionally, due to the YOLOv3's network modification, a new parameter is required to be set when training the model: the "loss weight." The new "network head" introduced by the semantic segmentation infusion has an independent loss function which its output is required to be combined with the YOLOv3's default loss value. This combination applies a weighting factor for each of the default and the infusion losses. According to the SDS-RCNN implementation, despite not reporting it, the authors do use a factor of 0.2 for the infusion head. That is, the loss value generated by the infusion head has only $\frac{1}{5}$ of the importance compared to the loss value produced by the YOLOv3 loss function. Instead of simply using the same factor, we decided to try different values for the different datasets in order to choose the factor based on some clue. Tables 7.24 and 7.25 present the experiments elaborated with different loss weights for the infusion head in different dataset configurations and for both YOLOv3 and YOLOv3-tiny models. The best "loss weight" for each combination of model and dataset has been manually chosen according to the general best performance when analyzing all the metrics. For the Caltech dataset, the Reasonable metric has been prioritized, while for the PTI01 the mAP has been chosen for reference. The selected weights are displayed in Table 7.26.

7.2.2 Results

Table 7.27 presents the results from the experimentation which compare the YOLOv3 and YOLOv3-tiny models between its original network and the one with the implementation of the infusion technique. The green cells in the table show the cases where the infusion has been effective in improving the model's accuracy. The red cells highlight the experiments which have the statistical decrease in the detection quality caused by the use of the infusion. For each infusion model, the "Null Hypothesis" (1) is evaluated, where rejecting it means there is an substantial effect when applying infusion, either negative or positive.

Null Hypothesis 1 *There is no significant difference in accuracy between the YOLOv3 models that make use of the infusion technique and the ones that do not.*

According to the table of results (Table 7.27), for the PTI01 dataset, there was no experiments that demonstrated statistical improvements in accuracy when using the infusion.

Table 7.24: Comparison of five different loss weights for the infusion head on YOLOv3 and YOLOv3-tiny in four different dataset configurations. The metrics mAP-50 and LAMR (Reasonable) are utilized to present the results. For both metrics, two scores are presented. The first is the best mAP-50 or LAMR (Reasonable) found over all epochs and the second is average of the best 5 of each kind of metric over all epochs. The mAP-50 for the Caltech1× Sanitized considers also testing on a sanitized set.

		Caltech10×		**Caltech1× Sanitized**	
Metric	**Loss weight**	YOLOv3	YOLOv3-tiny	YOLOv3	YOLOv3-tiny
Best mAP-50	0.2	0.3596	0.2091	0.5244	0.3158
	0.6	0.3353	0.2053	0.5392	0.3224
	1	0.3239	0.2049	0.5465	0.3132
	2	0.3095	0.1986	0.5547	0.3050
	4	0.3354	0.2144	0.5183	0.3193
	8	0.3296	0.1949	0.5720	0.3271
Avg. Top 5 mAP-50	0.2	0.3188	0.2008	0.5145	0.2961
	0.6	0.3130	0.1889	0.5177	0.2888
	1	0.3133	0.1934	0.5218	0.2991
	2	0.3015	0.1934	0.5325	0.2880
	4	0.3115	0.1948	0.5024	0.3048
	8	0.3114	0.1905	0.5450	0.3027
Best Reasonable	0.2	0.1850	0.4109	0.2070	0.4318
	0.6	0.2115	0.3999	0.1776	0.4575
	1	0.2420	0.4301	0.1825	0.4065
	2	0.2381	0.4247	0.1940	0.4289
	4	0.2473	0.3788	0.1772	0.4553
	8	0.2492	0.4371	0.1696	0.4703
Avg. Top 5 Reasonable	0.2	0.2487	0.4297	0.2279	0.4719
	0.6	0.2520	0.4334	0.2176	0.4975
	1	0.2630	0.4435	0.2118	0.4465
	2	0.2602	0.4259	0.2094	0.4778
	4	0.2650	0.4282	0.2092	0.4895
	8	0.2766	0.4527	0.1904	0.4824

Table 7.25: Comparison of five different loss weights for the infusion head on YOLOv3 and YOLOv3-tiny in two PTI01 dataset configurations. The mAP metric is evaluated using the 40% of IoU parameter (mAP-40). Two mAP-40 scores are presented. The first is the best mAP-40 found over all epochs and the second is average of the best five mAP-40 over all epochs.

		Leaving events out		**Leaving 3 cameras out & discarding**	
Metric	**Loss weight**	YOLOv3	YOLOv3-tiny	YOLOv3	YOLOv3-tiny
Best mAP-40	0.2	0.9022	0.7632	0.8339	0.6684
	0.6	0.9088	0.7613	0.8172	0.6630
	1	0.9141	0.7589	0.8153	0.6884
	2	0.9037	0.7615	0.8346	0.6846
	4	0.9094	0.7636	0.8351	0.6626
	8	0.9145	0.7591	0.8377	0.7010
Avg. Top 5 mAP-40	0.2	0.8990	0.7507	0.8264	0.6593
	0.6	0.9004	0.7444	0.8062	0.6498
	1	0.9055	0.7498	0.8042	0.6660
	2	0.9009	0.7456	0.8284	0.6742
	4	0.9029	0.7461	0.8151	0.6475
	8	0.9082	0.7538	0.8176	0.6668

Table 7.26: Loss weights that have been chosen for the infusion experimentation.

	YOLOv3	**YOLOv3-tiny**
Caltech10×	0.2	4
Caltech1× Sanitized	8	1
PTI01 leaving events out	2	2
PTI01 leaving 3 cameras out & discarding	2	2

Curiously, a single experiment demonstrated to be negative affected by infusion: the YOLOv3 model for the "events out" configuration using the LAMR metric. The Null Hypothesis is statistically accepted for the remaning of the experiments (white cells), thus it means the infusion

did not statistically decrease or increase the model's accuracy in such cases. For the PTI01 dataset, when evaluating using the mAP metric, the scores are very similar when comparing the experiments with and without infusion. However, when checking the results evaluated with LAMR, even that there is no statistical differences, it is possible to verify a small change in accuracy. For the "events out" configuration, the LAMR demonstrates a decrease in the score for the YOLOv3-tiny as well. Therefore, the infusion did prejudice the model when training with the dataset in the "events out" configuration. For the "cameras out" we observe the contrary: both YOLOv3 and YOLOv3-tiny models demonstrates slight improvements in accuracy for the LAMR between 0.5% and 2%, although it is not statistical.

Despite the not presenting positive statistical effects on the PTI01 experiments, the infusion has demonstrated better results in the Caltech dataset. Observing the "Caltech1× Sanitized" (Zhang et al., 2016b), the infusion has been statistically effective for all experiments but one. The result where the Null Hypothesis has been accepted refers to the YOLOv3-tiny model evaluated by the "Avg. mAP" metric. Regardless it did not achieved enough significance in the differences of the means, the accuracy has increased in 1% comparing the default tiny model (0.173±0.01, n=20) and the one with infusion (0.1832±0.01, n=20). Finally, "Caltech10×" (Dollár et al., 2009b) results demonstrate that the infusion has been less effective in this dataset compared to the sanitized version. While YOLOv3-tiny rejected the Null Hypothesis for every metric, YOLOv3 did accept it in all of them. In the "Best Reasonable" metric, YOLOv3 with infusion improved the accuracy in 1.16% (0.1996±0.02, n=20) compared to the default model (0.2112±0.02, n=20).

Table 7.27: Results for the YOLOv3 using the "weak semantic segmentation infusion" technique. Both YOLOv3 and YOLOv3-tiny models are tested over four different dataset configurations: "Caltech10×" (Dollár et al., 2009b), "Caltech1× Sanitized" (Zhang et al., 2016b), "PTI01 leaving events out" (PTI01 Events Out), and "PTI01 leaving 3 cameras out & discarding 33% of training data" (PTI01 Cameras Out). The PTI01 datasets are evaluated using the mAP-40 and LAMR metrics for the best-scored epoch of each model (Best) and for the top 5 best-scored epochs of each model (Avg). The Caltech datasets are evaluated using the mAP-50 and the LAMR metric using the "Reasonable" setting, also considering the best epoch and top 5 epochs scheme. Each score in the table is an averaged value of 10 models accompanied by the standard deviation. In total, this table reports the results of 160 trained models. For the infusion models (inf), a conclusion based on the "T Test" statistical test is presented as the "(A)" tag if the results accept the Null Hypothesis that there is no significant difference between the original model and the respective one with infusion. If the statistical test finds that the score has a meaningful difference the "(R)" tag is used to represent the rejection of the Null Hypothesis. The statistical improvements are highlighted in green while the decreases are in red.

Dataset	Model	Best mAP	Avg. mAP	Best LAMR	Avg. LAMR
Caltech10x	YOLOv3	0.352±0.02	0.3137±0.01	0.2112±0.02	0.2563±0.02
	YOLOv3 (Inf.)	0.3526±0.02(A)	0.3098±0.01(A)	0.1996±0.02(A)	0.2594±0.02(A)
	YOLOv3-tiny	0.1969±0.02	0.1848±0.01	0.4295±0.02	0.4462±0.01
	YOLOv3-tiny (Inf.)	0.2109±0.02(R)	0.1964±0.01(R)	0.4003±0.02(R)	0.4257±0.02(R)
Caltech1x Sanitized	YOLOv3	0.3502±0.02	0.333±0.01	0.2103±0.02	0.2337±0.02
	YOLOv3 (Inf.)	0.3645±0.02(R)	0.3471±0.01(R)	0.1836±0.02(R)	0.2135±0.02(R)
	YOLOv3-tiny	0.1891±0.01	0.173±0.01	0.4688±0.02	0.5154±0.02
	YOLOv3-tiny (Inf.)	0.2±0.02(R)	0.1832±0.01(A)	0.4296±0.03(R)	0.4765±0.03(R)
PTI01 Events Out	YOLOv3	0.9068±0.01	0.9008±0.01	0.1464±0.01	0.1554±0.01
	YOLOv3 (Inf.)	0.9054±0.01(A)	0.8982±0.01(A)	0.1566±0.02(R)	0.1672±0.01(R)
	YOLOv3-tiny	0.7572±0.01	0.7422±0.01	0.3879±0.01	0.4076±0.02
	YOLOv3-tiny (Inf.)	0.7597±0.01(A)	0.7455±0.01(A)	0.3913±0.02(A)	0.4146±0.02(A)
PTI01 Cameras Out	YOLOv3	0.8251±0.02	0.8133±0.01	0.2409±0.02	0.2612±0.01
	YOLOv3 (Inf.)	0.828±0.01(A)	0.8182±0.01(A)	0.2376±0.02(A)	0.2587±0.03(A)
	YOLOv3-tiny	0.6771±0.02	0.6535±0.02	0.4515±0.02	0.4825±0.03
	YOLOv3-tiny (Inf.)	0.6756±0.02(A)	0.6592±0.02(A)	0.4353±0.03(A)	0.4607±0.03(A)

Table 7.28 compares the bests trained models on Caltech which are part of the results presented in Table 7.27. It is possible to verify that the models that use infusion enable an increase

of the accuracy in the majority of the LAMR metrics. For instance, The YOLOv3-tiny with infusion decreases the LAMR for the Reasonable setting in 6.34%. Also, despite the YOLOv3 with infusion not statistically achieving significant general improvements over the default model, the infusion does help with atypical aspect ratios (Ar=atypical in 3.19%), with heavy occlusion (Occ=heavy in 2.25%), and with large-scaled pedestrians (Scale=large in 2.82%). This effect increases when evaluating the Caltech1× Sanitized and also when using the YOLOv3-tiny model. For instance, the infusion applied to YOLOv3-tiny on Caltech10× helps decreasing the LAMR for heavy occlusions in 10.45%, while the YOLOv3 trained with infusion does help with 11.02% in the same metric but for the Caltech1× Sanitized.

Figure 7.22 demonstrates some improvements provided by the application of the infusion technique on the YOLOv3 model trained on the Caltech1× Sanitized dataset. Some False Negative annotations are presented for the model without infusion corresponding to pedestrians missed by the detector, while the YOLOv3 with infusion demonstrates to correctly detect those cases.

Table 7.28: The LAMR metric evaluates the YOLOv3 and YOLOv3-tiny with and without the infusion technique on two Caltech dataset configurations: Caltech10× (Dollár et al., 2009b) and Caltech1× (Zhang et al., 2016b). The Caltech1× is tested on the sanitized set. The "Difference" value shows the percentile of dissimilarity of the best YOLOv3 model compared to the best YOLOv3 with infusion (including the "tiny" version). The selected best models are from the same test batch reported in Table 7.27.

Dataset	Model	Reasonable	All	Ar=all	Ar=atypical	Ar=typical	Occ=heavy
Caltech10×	YOLOv3	0.1853	0.5747	0.1621	0.2346	0.1477	0.6190
	YOLOv3 (inf)	0.1805	0.5719	0.1552	0.2027	0.1456	0.5965
Difference		*0.47%*	*0.28%*	*0.69%*	*3.19%*	*0.21%*	*2.25%*
Caltech1× Sanitized	YOLOv3	0.1906	0.5991	0.1702	0.2836	0.1442	0.5874
	YOLOv3 (inf)	0.1539	0.5667	0.1341	0.2398	0.1133	0.4772
Difference		*3.68%*	*3.24%*	*3.62%*	*4.38%*	*3.09%*	*11.02%*
Caltech10×	YOLOv3-tiny	0.4022	0.7560	0.3702	0.4573	0.3507	0.8625
	YOLOv3-tiny (inf)	0.3738	0.7452	0.3487	0.4286	0.3277	0.7580
Difference		*2.85%*	*1.08%*	*2.15%*	*2.87%*	*2.30%*	*10.45%*
Caltech1× Sanitized	YOLOv3-tiny	0.4394	0.7888	0.4209	0.4790	0.3980	0.8047
	YOLOv3-tiny (inf)	0.3760	0.7455	0.3544	0.4394	0.3313	0.7546
Difference		*6.34%*	*4.33%*	*6.65%*	*3.96%*	*6.68%*	*5.01%*

Dataset	Model	Occ=none	Occ=partial	Scale=far	Scale=large	Scale=medium	Scale=near
Caltech10×	YOLOv3	0.1621	0.3799	0.8187	0.0691	0.4468	0.0622
	YOLOv3 (inf)	0.1552	0.3661	0.8109	0.0409	0.4435	0.0767
Difference		*0.69%*	*1.38%*	*0.78%*	*2.82%*	*0.33%*	*-1.45%*
Caltech1× Sanitized	YOLOv3	0.1702	0.3200	0.8895	0.0739	0.4821	0.0882
	YOLOv3 (inf)	0.1341	0.2940	0.8990	0.0552	0.4402	0.0753
Difference		*3.62%*	*2.60%*	*-0.95%*	*1.86%*	*4.19%*	*1.29%*
Caltech10×	YOLOv3-tiny	0.3702	0.6518	0.9635	0.2279	0.6737	0.2316
	YOLOv3-tiny (inf)	0.3487	0.5539	0.9570	0.1437	0.6747	0.1683
Difference		*2.15%*	*9.78%*	*0.65%*	*8.42%*	*-0.10%*	*6.34%*
Caltech1× Sanitized	YOLOv3-tiny	0.4209	0.5352	0.9905	0.1860	0.7387	0.2499
	YOLOv3-tiny (inf)	0.3544	0.5144	0.9861	0.1938	0.6716	0.2255
Difference		*6.65%*	*2.08%*	*0.44%*	*-0.78%*	*6.71%*	*2.45%*

7.2.2.1 The best "YOLOv3 + Infusion" model against the top tier methods on Caltech

Similarly to Section 7.1.8.2, in this experiment we change the default training parameters in order to reach the best accuracies using the YOLOv3 model with the infusion technique. In the previous sections, the infusion experiments have made use of parameters of faster convergences, but, probably not generating the best possible scores. To achieve such results, we change the learning rate (1^{-5}), anchors (k=3 with our recalculation), and data augmentation (default method) in the same way as it has been done in Section 7.1.8.2. The metrics are the mAP-50 and the LAMR, and the evaluated dataset is the Caltech10×.

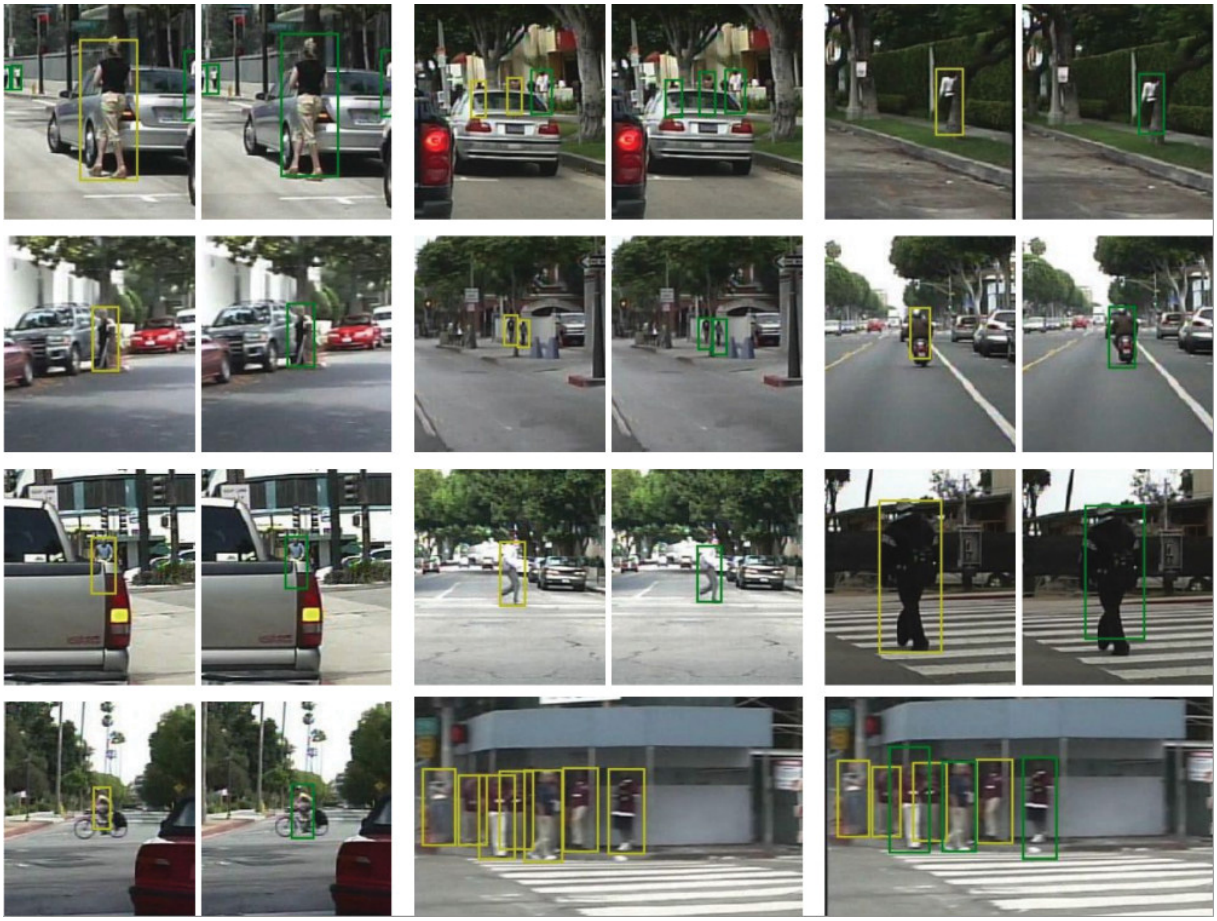


Figure 7.22: Comparing the detection of the best YOLOv3 model trained and tested on Caltech 1× Sanitized (Zhang et al., 2016b) according to Table 7.28. Each image on the left focuses on presenting the False Negative detections (yellow rectangles) provided by the YOLOv3 without infusion, that is, pedestrians which have been missed by the detector. On the right, the YOLOv3 with infusion demonstrates improvements in detection (True Positives in green rectangles).

According to Table 7.29, the best YOLOv3 model with the "weak semantic segmentation" infusion technique achieved better performance in the majority of the metrics compared to the model without the technique. The model could not defeat its original version in the "Large Scale," "Near Scale" and "Heavy Occlusion" settings. Even so, the current model did better than the SDS-RCNN (Brazil et al., 2017) in the "Heavy Occlusion", and stayed only 2.94% behind it in the main setting (Reasonable). Compared to the original model (without infusion), the current model enabled the achievement of better results in the "AR Atypical" against the SDS-RCNN, "Partial Occlusion" against TLL-TFA (Song et al., 2018), and "Far Scale" against GDFL (Lin et al., 2018).

The YOLOv3 with Infusion demonstrates general improvements over the evaluated settings. However, somehow, the infusion technique degraded the detection quality of pedestrians in the biggest scales, as well for the ones which are severed occluded. For the first time, the YOLOv3 could do better than TLL-TFA in a metric setting, the "Partial Occlusion". According to the current Caltech Pedestrian Detection benchmark on the Reasonable setting, the YOLOv3 would fit in the 12th position in the ranking, while the YOLOv3 without infusion would be in the 15th (Figure 7.23).

Table 7.29: Comparing the best YOLOv3 model with the infusion technique against the three top methods in the Caltech's official benchmark and the best YOLOv3 model without infusion (Redmon and Farhadi, 2018). The compared methods are SDS-RCNN (Brazil et al., 2017), TLL-TFA (Song et al., 2018) and GDFL (Lin et al., 2018). Twelve different configurations for the LAMR metric are reported according to the Caltech's benchmark tool. The YOLOv3 with Infusion (YOLOv3 Inf.) score and IoU inference thresholds are set as 0.01 and 0.5, respectively. The lower the metric value, the better the performance. Additionally, the mAP-50 metric is presented, where the higher the value, the better. The "AR" stands for Aspect Ratio and the "Occl." for occlusion. Values in bold are the best method in the metric. The green values highlight the metrics where the YOLOv3 with Infusion wins over the orange marked methods.

	Methods				
Metric	SDS-RCNN	TLL-TFA	GDFL	YOLOv3	YOLOv3 Inf.
Reasonable	**7.36%**	7.40%	7.85%	12.85%	10.30%
Overall	61.50%	**38.15%**	48.14%	53.11%	49.83%
AR All	5.95%	**5.85%**	6.36%	11.47%	9.27%
AR Atypical	11.68%	**9.09%**	14.67%	11.98%	11.46%
AR Typical	4.58%	5.09%	**4.50%**	11.01%	8.62%
No Occl.	5.95%	**5.85%**	6.36%	11.47%	9.27%
Partial Occl.	**14.86%**	18.50%	16.74%	20.71%	16.81%
Heavy Occl.	58.55%	**28.66%**	43.18%	49.56%	54.17%
Near Scale	2.15%	**0.72%**	1.69%	1.54%	3.09%
Medium Scale	50.88%	**22.92%**	32.50%	40.25%	34.34%
Far Scale	100%	**60.09%**	70.97%	71.68%	69.39%
Large Scale	0.97%	**0.00%**	1.14%	**0.00%**	1.36%
mAP-50	-	-	-	45.76%	**48.3%**

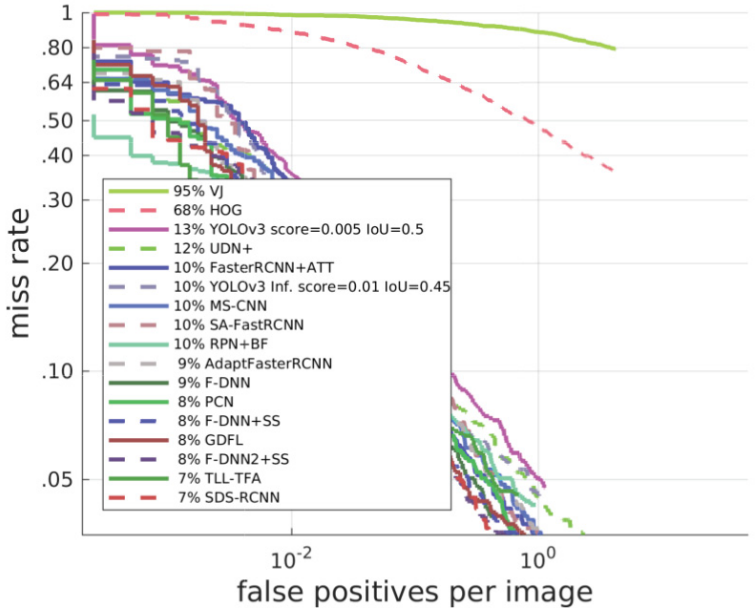


Figure 7.23: Official Caltech Pedestrian Benchmark with the thirteen best methods, with the additional VJ and HOG methods for reference. According to the results achieved in the Reasonable setting for the LAMR metric, the YOLOv3 reached the 15th position with a score of 12.85% while the YOLOv3 with Infusion conquered the 12th position with a score of 10.30%. The ranking was collected from the official repository[2] on 01/15/2019.

7.2.3 General Conclusions

The "weak semantic segmentation infusion" demonstrated the best results in the Caltech dataset, specially on the Caltech1× Sanitized (Zhang et al., 2016b) configuration. Although, it did prejudice the model in a single configuration on the PTI01, while not helping in the remaining of the experiments.

Despite not having enough data for a solid conclusion about the poorer effect on PTI01, we create the hypothesis to be solved in future works. We theorize that the infusion may positively

affect the model at a certain level of difficulty. In other words, our theory is that YOLOv3 and YOLOv3-tiny have enough learning power in the PTI01 dataset so that the infusion does not provide any significant additional information about the pedestrian locations. Therefore, when working on a much harder dataset (Caltech), the models suffer from such adverse data, thus the information given by the semantic segmentation infusion starts to get relevant. This theory may be enforced by the fact that the YOLOv3-tiny did benefit more than the more complex YOLOv3 model (the default). This effect can also be verified in Table 7.28 which compares the best models with and without infusion. The smaller model has less learning power compared to the bigger one and may benefit from any additional source of useful information.

We have elaborated an experiment on the PTI01 dataset trying to acquire additional clues to check that theory. For that, it was used the PTI01 with the configuration "leaving 3 cameras out" while discarding 66% of the training data. The initial idea was to create a harder scenario which would make the YOLOv3 model to find useful any additional features coming from the infusion. According to the results presented in Table 7.13, the evaluations using the mAP did not show relevant differences when using infusion. However, observing the LAMR scores, it was possible to see that for the easier dataset configuration ("events out") the infusion did prejudice the model's accuracy. This may infer that the infusion weight chosen for new head could be too high, so that, the infusion clues were disturbing the normal detection features. The problem may have been related to the fact that we have chosen the mAP metric instead of the LAMR when selecting the weights. In the other hand, the hard configuration ("cameras out") enabled the models to make use of the infusion, slightly increasing the detection quality. These results indicates that our theory may be right. Even so, future works are important in order to provide a solid explanation about such behaviors.

By fact, the different performances achieved by the models are very related to the dataset configurations they have been submitted. Because of that, it is possible to conclude that the infusion technique has a positive effect on both YOLOv3 and YOLOv3-tiny versions depending on the scenario where it is trained and tested. The only solid clues are that the infusion did perform better on a lower complexity model and harder datasets.

According to the results, it is possible to verify that the mAP metrics punish much more the model's performance compared to the LAMR (Reasonable) when comparing their proportions in the Caltech configurations. Using the mAP metric as a reference, it is also possible to understand that PTI01 represents an easier challenge for both YOLOv3 and YOLOv3-tiny models according to the superior scores compared to the Caltech datasets. Also, the mAP did not reflect the decreases and increases in the accuracy which have been detected by the LAMR metric.

In Section 7.2.2.1, we see that applying a finer parameterization for training YOLOv3, with the additional modification of the inference parameters (score and IoU thresholds) enabled the models to achieve greater accuracies in both metrics (mAP and LAMR). Such results placed the YOLOv3 with Infusion in the 12th position in the Caltech's benchmark ranking. Also, the new parameters accentuated the enhancement provided by the infusion technique which the "T-Test" statistical test had initially marked the improvement as insufficient, by accepting the Null Hypothesis.

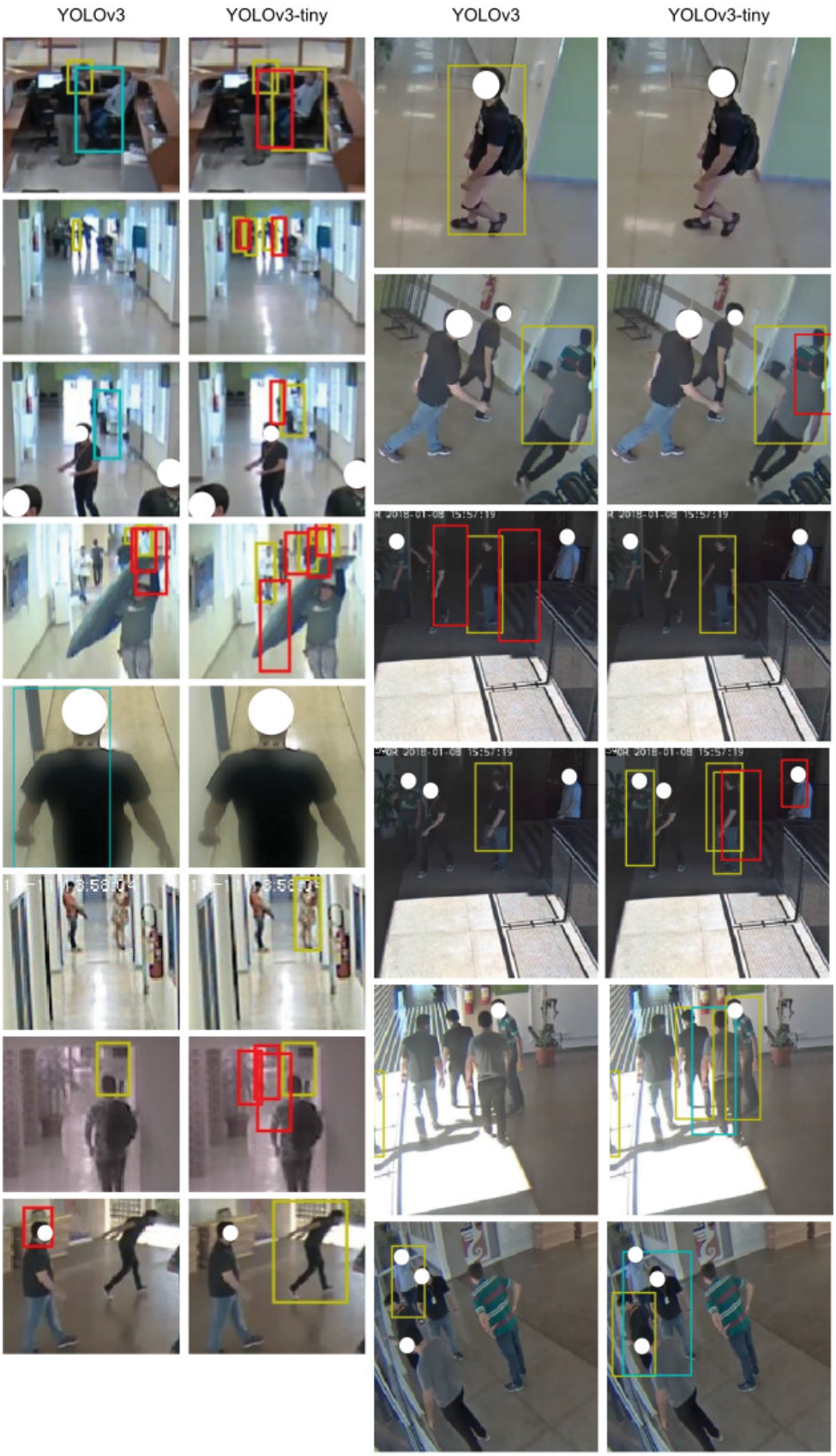


Figure 7.24: Prediction samples from YOLOv3 and YOLOv3-tiny over PTI01 dataset. The yellow bounding boxes are False Negatives (undetected pedestrians). The red boxes are False Positives, and the blue ones are False Positives by Multiple Detection. Ground truths and True Positives are not displayed to make the visualization cleaner. The pedestrians identities are protected by a white mask on the head.

8 CONCLUSION

The Pedestrian Detection aims in locating pedestrians in images and has been considered essential in many computational tasks which can improve the quality of life (Dollár et al., 2012), for activities like driving assistance, video surveillance, human interfaces, autonomous robots, and vehicles. The Computer Vision area has been successfully tackling the Pedestrian Detection with Deep Convolutional Neural Networks (DCNNs) powered by the increasing computational performances provided by the Graphics Processing Units (GPUs).

Despite the continuous progress in the task, it is not solved yet. According to Liu et al. (2018), Pedestrian Detection has a long way of improvements before reaching satisfactory levels, or human performance (Zhang et al., 2016b). It has been considered one of the most challenging problems in this computational area (Wang et al., 2018), where such problems are mostly related to variations in the image like illumination, people's clothing, and articulation of the human body, as well the high processing costs.

The Pedestrian Detection, as a specialized object detector, seeks to accomplish two main goals: high accuracy and high efficiency (Liu et al., 2018). The first tries to improve the quality of the detections, while the efficiency looks for executing the tasks with the minimum effort. Unfortunately, progresses in accuracy are typically related to the usage of progressive, complex solutions which infers to elevated computational costs (Dollár et al., 2010; Guo et al., 2017). On the other side, the detection speed has been considered as important as the accuracy (Varga and Szirányi, 2017). The speed has also been defined as a fundamental quality for applications like robotics, automotive safety, and surveillance (Dollár et al., 2010).

Many Pedestrian Detectors which are part of the most accurate ones in some challenges, such as SDS-RCNN (Brazil et al., 2017) on the Caltech Pedestrian Benchmark, are based on heavy neural network models like the two-stage region based methods (e.g., Faster R-CNN (Ren et al., 2015)). Nevertheless, there are detectors known by being lighter and by consequence, faster. The latter type is commonly composed of unified pipelines, such as the YOLO (Redmon and Farhadi, 2018). YOLOv3 has proven to be the fastest detector in its main challenge (COCO, Lin et al. (2014)), while sustaining the best or competitive accuracies, depending on the metric.

Curiously, we have not been able to find works reporting the performance of YOLOv3 on the Caltech Benchmark. Because of that, this work experiment the YOLO method trying to measure its native "accuracy × speed" trade-off. YOLOv3 is not a natural Pedestrian Detector, but a Generic Object Detector, which deals with 80 classes of objects in the COCO challenge. This enhanced our interest in checking its performance after a possible specialization for pedestrians. In order to do that, we developed an exploration of the YOLO over many different environments and configurations, aiming to find ways to tune the model in a sufficient form to compete in the Caltech Benchmark.

Beyond the Caltech dataset, due to the importance of the pedestrian detection on video surveillance (Du et al., 2017; Varga and Szirányi, 2017; Ye et al., 2016; Zheng et al., 2016a), we have created an additional dataset, called "PTI01", from a realistic video monitoring network. This dataset enabled the evaluation of the YOLOv3 in a much more meaningful scenario, once the existing datasets available had a high discrepancy to the surveillance system from the Itaipu Technological Park (PTI), where the dataset has been built. The dataset is composed of almost 8,000 images from 21 cameras containing annotations of more than 30,000 pedestrians, while the PTI's video surveillance network is formed by more than 250 cameras generating more than 17,000,000 images per day, which record approximately 6,000 individuals during the year.

According to the achieved results, the PTI01 dataset provided an easier pedestrian detection task compared to the Caltech and a more agile experimentation setup due to its faster training time. Both characteristics turned the PTI01 the primary dataset for general experimentation. The Caltech is known by being big, taking 2.5 days to train a single YOLOv3 model on it, while when training on PTI01 it would take about 10 hours. Also, some problems in the annotations of the Caltech have demonstrated to affect the training stability. Therefore, after finding the right directions for the model on PTI01, the experiments elaborated on Caltech have explored a filtered range of options, saving exploration-time.

During the process of exploration, we have been able to discard some techniques initially theorized as effective, like the "canonical bounding boxes." Also, we modified some YOLO procedures to improve the detection quality, for instance, by using our simpler implementation of a clusterization algorithm to provide better "anchor boxes," which are a component of the YOLO. We have also compared the YOLOv3 with its lower-complexity network version, called YOLOv3-tiny, which demonstrated to have impressive accuracy close to the much heavier default version (getting only 6% behind in some experiments) and being extremely fast (169 FPS against 34 of the default model).

We have verified by comparative reasoning that the YOLOv3 can find more pedestrians in the same period compared to the top-3 methods in the Caltech Benchmark due to its competitive accuracy and faster detection speed. However, YOLOv3 demonstrated to provide a higher number of False Positives, which depending on the application may or may not affect the task. In a hypothetical scenario based on the PTI's video surveillance network, we have estimated that YOLOv3-tiny could handle, with almost a single instance, the 250 cameras in real-time. The YOLOv3 default model would require about five instances and the SDS-RCNN about 25 instances. Such analysis enables the understanding of the real-world feasibility of a pedestrian detector and confirms that detection speed is so important as the detection quality. A detector which perfectly detects the pedestrians but is too slow, may not be applied in a real case due to excessive resources demanded. YOLOv3 has shown lower accuracy but better speed, and it may fit better for the practical application compared to the best-scored model.

At the end of the exploration, we have been able to tune the YOLOv3 on Caltech, achieving a LAMR score of 12.85% on the main setting ("Reasonable"), staying behind the best method (SDS-RCNN, Brazil et al. (2017)) by 5.49%. Despite not beating the SDS-RCNN on the main setting, from 12 settings evaluated, YOLOv3 was able to overcome it in 6 of them, including the "Heavy occlusion" (pedestrians severely occluded), "Far Scale" (small-scaled pedestrians) and "Overall" (all the annotations). We found this result as very promising, once putting more effort on tuning the YOLOv3 could lead to the achievement of greater performances on the challenge.

In our last contribution, we have applied a technique called "weak semantic segmentation infusion" to the YOLOv3's neural network. Our inspiration came from the SDS-RCNN which applied the same technique and reported the increase in accuracy to the pedestrian detection, while not demanding additional processing power to generate the predictions. Such an approach fits well to contradict the tendency of the need to increase the accuracy by increasing the computational costs. The objective was to implement the method in the YOLO's network without affecting its speed performance. According to statistical tests, we have verified that the infusion technique did not provide benefits to the YOLOv3 model while training it in the PTI01 dataset. However, it has shown significant improvements in accuracy in the Caltech dataset, mainly for the YOLOv3-tiny model. We theorized that the infusion might be beneficial to the YOLOv3 model only when the dataset is harder than a certain level, which we could not find it out. The clues are related to the difference in the model's behavior in different dataset difficulties, as well

in the model's complexity. The infusion demonstrated to take better effect on a lower-complexity model (YOLOv3-tiny), and on harder dataset configurations. Significant increase in performance has been obtained on the sanitized version of the Caltech (Zhang et al., 2016b) which has higher quality annotations compared to the default version (Dollár et al., 2009b).

At the end, making use of the "weak semantic segmentation infusion," we have been able to place the YOLOv3 in the 12th position of the Caltech Pedestrian Detection Benchmark ranking. The modified model overcame its original version on 9 out of the 12 settings. Compared to the SDS-RCNN, the YOLOv3 with infusion stays only 2.94% behind in the main setting (Reasonable). These results demonstrate the effectiveness of the infusion technique on YOLOv3 but open some questions about its performance on different datasets which should be explored in future works.

8.1 FUTURE WORKS

According to the Pedestrian Detection context presented in this work, we find essential for the further evolution of the achieved results, at least, the following studies:

- A more-in-depth exploration of YOLOv3 on Caltech, trying to improve the currently achieved accuracy, mainly by finding finer and optimal training parameters. That includes trying to understand why the accuracy keeps dropping after each trained epoch for the Caltech10×;
- Modify the YOLOv3 model and algorithms to work with TLL annotations instead of using bounding boxes. They have been successfully utilized by the TLL-TFA method (Song et al., 2018), which holds the best score on the majority of the metrics evaluated on Caltech. This could dramatically increase the YOLO's performance. The impact of detection speed should also be carefully analyzed once TLL-TFA does not report any information in this regard;
- Modify the YOLOv3 neural network to include some Long-short Term Memory (LSTM) technique. For instance, the TLL-TFA (Song et al., 2018) demonstrates improvements in accuracy for detecting pedestrians in sequential frames, which is the case for the Caltech and PTI01 datasets. However, the LSTM may not take effect for detection in random images, and the inference speed may be negatively affected due to the extra processing costs;
- Modify the YOLOv3 neural network trying to increase its detection speed by, for instance, removing the number of filters in each layer. The idea is to make the model lighter without significantly impacting the detection quality. YOLOv3 has been prepared for a challenge of 80 classes, and because of that, when detecting a single person, the reduction of the network size may be proportionally beneficial. This is expected due to the great performance demonstrated by the YOLOv3-tiny in our experimentation;
- Find a way to apply the semantic segmentation clues on the YOLOv3's network, as have been done by GDFL (Lin et al., 2018) where they achieved great accuracy and speeds;
- Create a new version of the PTI01 dataset including more frames, scene variety, and TLL annotations;

- Check our hypothesis that the "weak semantic segmentation infusion" may work only in very hard scenarios. The idea is to explain why the infusion technique did not work on the PTI01 dataset, and hopefully find a measurement which would help in deciding when the infusion may or may not be worth it;
- Try to create standardized bounding boxes through the YOLOv3's predictions in the same way reported by the Dollár et al. (2009b) verifying the improvements in accuracy. Despite not reporting it, we have tried to execute such approach achieving worse results, but without a solid basement to prove the method as inefficient for the YOLOv3;
- Compare the YOLOv3's performance against YOLOv2 on Caltech.

References

Baba, A., Pasha, M. G., Ahammed, S. A., and Tabassum, S. N. (2013). Introduction to Neural Networks Design Architecture. *International Journal of Scientific & Engineering Research*, 4(2):1–8.

Bedagkar-Gala, A. and Shah, S. K. (2014). A survey of approaches and trends in person re-identification. *Image and Vision Computing*, 32(4):270–286.

Benenson, R., Omran, M., Hosang, J., and Schiele, B. (2014). Ten Years of Pedestrian Detection, What Have We Learned? In *European Conference on Computer Vision*, pages 613–627. Springer.

Bezerra, C. S., Laroca, R., Lucio, D. R., Severo, E., Oliveira, L. F., Britto Jr., A. S., and Menotti, D. (2018). Robust iris segmentation based on fully convolutional networks and generative adversarial networks. In *Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 281–288.

Brazil, G., Yin, X., and Liu, X. (2017). Illuminating Pedestrians via Simultaneous Detection and Segmentation. *2017 IEEE International Conference on Computer Vision (ICCV)*.

Cai, Z., Saberian, M., and Vasconcelos, N. (2015). Learning Complexity-aware Cascades for Deep Pedestrian Detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3361–3369.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223.

Correia, A. J. L. (2016). The Good, the Fast and the Better Pedestrian Detector. Master's thesis, Federal University of Minas Gerais.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE.

Deng, L., Yu, D., et al. (2014). Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387.

Dollár, P., Belongie, S. J., and Perona, P. (2010). The Fastest Pedestrian Detector in the West. In *Bmvc*, volume 2, page 7. Citeseer.

Dollár, P., Tu, Z., Perona, P., and Belongie, S. (2009a). Integral channel features. *British Machine Conference*.

Dollár, P., Wojek, C., Schiele, B., and Perona, P. (2009b). Pedestrian Detection: A Benchmark. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 304–311. IEEE.

Dollár, P., Wojek, C., Schiele, B., and Perona, P. (2012). Pedestrian Detection: An Evaluation of the State of the Art. *IEEE transactions on pattern analysis and machine intelligence*, 34(4):743–761.

Du, X., El-Khamy, M., Lee, J., and Davis, L. (2017). Fused DNN: A deep neural network fusion approach to fast and robust pedestrian detection. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pages 953–961. IEEE.

Du, X., El-Khamy, M., Morariu, V. I., Lee, J., and Davis, L. (2018). Fused Deep Neural Networks for Efficient Pedestrian Detection. *arXiv preprint arXiv:1805.08688*.

Duan, L., Lou, Y., Wang, S., Gao, W., and Rui, Y. (2017). AI Oriented Large-Scale Video Management for Smart City: Technologies, Standards and Beyond. *arXiv preprint arXiv:1712.01432*.

Enzweiler, M. and Gavrila, D. M. (2009). Monocular Pedestrian Detection: Survey and Experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2179–2195.

Ess, A., Leibe, B., and Van Gool, L. (2007). Depth and Appearance for Mobile Scene Analysis. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The PASCAL Visual Object Classes (VOC) Challenge. *International journal of computer vision*, 88(2):303–338.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object Detection with Discriminatively Trained Part-Based Models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645.

Fu, C., Liu, W., Ranga, A., Tyagi, A., and Berg, A. C. (2017). DSSD : Deconvolutional Single Shot Detector. *CoRR*, abs/1701.06659.

Gajjar, V., Gurnani, A., and Khandhediya, Y. (2017). Human Detection and Tracking for Video Surveillance: A Cognitive Science Approach. *arXiv preprint arXiv:1709.00726*.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Guo, Z., Wu, F., Chen, H., Yuan, J., and Cai, C. (2017). Pedestrian violence detection based on optical flow energy characteristics. In *Systems and Informatics (ICSAI), 2017 4th International Conference on*, pages 1261–1265. IEEE.

Hagan, M. T., Demuth, H. B., Beale, M. H., and De Jesús, O. (1996). *Neural network design*, volume 20. Pws Pub. Boston, 2 edition.

He, K., Gkioxari, G., Dollar, P., and Girshick, R. (2017). Mask R-CNN. *2017 IEEE International Conference on Computer Vision (ICCV)*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hu, Q., Wang, P., Shen, C., van den Hengel, A., and Porikli, F. (2018). Pushing the Limits of Deep CNNs for Pedestrian Detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(6):1358–1368.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE CVPR*.

Huang, S. and Ramanan, D. (2017). Expecting the Unexpected: Training Detectors for Unusual Pedestrians with Adversarial Imposters. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1.

Junlin, H., Jiwen, L., Yap-Peng, T., and Jie, Z. (2016). Deep Transfer Metric Learning. *IEEE transactions on image processing: a publication of the IEEE Signal Processing Society*, 25(12):5576–5588.

Kim, T. K. (2015). T test as a parametric statistic. *Korean Journal of Anesthesiology*, 68:540–6.

Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Ko, K.-E. and Sim, K.-B. (2018). Deep convolutional framework for abnormal behavior detection in a smart surveillance system. *Engineering Applications of Artificial Intelligence*, 67:226–234.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Lai, J. and Maples, S. (2017). Developing a Real-Time Gun Detection Classifier. *Stanford University*.

Laroca, R., Barroso, V., Diniz, M. A., Gonçalves, G. R., Schwartz, W. R., and Menotti, D. (2019). Convolutional neural networks for automatic meter reading. *Journal of Electronic Imaging*, 28:1–14.

Laroca, R., Severo, E., Zanlorensi, L. A., Oliveira, L. S., Gonçalves, G. R., Schwartz, W. R., and Menotti, D. (2018). A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lin, C., Lu, J., Wang, G., and Zhou, J. (2018). Graininess-Aware Deep Feature Learning for Pedestrian Detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 732–747.

Lin, T., Dollár, P., Girshick, R. B., He, K., Hariharan, B., and Belongie, S. J. (2016). Feature Pyramid Networks for Object Detection. *CoRR*, abs/1612.03144.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollar, P. (2017). Focal Loss for Dense Object Detection. *2017 IEEE International Conference on Computer Vision (ICCV)*.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. *Lecture Notes in Computer Science*, page 740–755.

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., and Pietikäinen, M. (2018). Deep Learning for Generic Object Detection: A Survey. *arXiv preprint arXiv:1809.02165*.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). SSD: Single Shot Multibox Detector. In *European conference on computer vision*, pages 21–37. Springer.

Mao, J., Xiao, T., Jiang, Y., and Cao, Z. (2017). What Can Help Pedestrian Detection? In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 3.

Masters, D. and Luschi, C. (2018). Revisiting Small Batch Training for Deep Neural Networks.

Milan, A., Leal-Taixé, L., Reid, I., Roth, S., and Schindler, K. (2016). MOT16: A Benchmark for Multi-Object Tracking. *arXiv preprint arXiv:1603.00831*.

Mitchell, T. M. et al. (1997). Machine learning. WCB.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of machine learning*. MIT press.

Molchanov, V., Vishnyakov, B., Vizilter, Y., Vishnyakova, O., and Knyaz, V. (2017). Pedestrian detection in video surveillance using fully convolutional YOLO neural network. In *Automated Visual Inspection and Machine Vision II*, volume 10334, page 103340Q. International Society for Optics and Photonics.

Nielsen, M. A. (2015). *Neural Networks and Deep Learning*, volume 25. Determination press USA.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Papageorgiou, C. and Poggio, T. (2000). A Trainable System for Object Detection. *International journal of computer vision*, 38(1):15–33.

Patterson, J. and Gibson, A. (2017). *Deep Learning: A Practitioner's Approach*. " O'Reilly Media, Inc.".

Pitts, W. and McCulloch, W. S. (1947). How we know universals the perception of auditory and visual forms. *The Bulletin of mathematical biophysics*, 9(3):127–147.

Powers, D. M. (2011). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies*.

Rahman, M. A. and Wang, Y. (2016). Optimizing Intersection-Over-Union in Deep Neural Networks for Image Segmentation. In *International Symposium on Visual Computing*, pages 234–244. Springer.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

Redmon, J. and Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Redmon, J. and Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *CoRR*, abs/1804.02767.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. (2008). LabelMe: A Database and Web-Based Tool for Image Annotation. *International journal of computer vision*, 77(1-3):157–173.

Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 3(3):210–229.

Schmidhuber, J. (2015). Deep Learning in Neural Networks: An Overview. *Neural networks*, 61:85–117.

Sharma, V., Rai, S., and Dev, A. (2012). A Comprehensive Study of Artificial Neural Networks. *International Journal of Advanced research in computer science and software engineering*, 2(10).

Simonyan, K. and Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, abs/1409.1556.

Song, T., Sun, L., Xie, D., Sun, H., and Pu, S. (2018). Small-scale Pedestrian Detection Based on Somatic Topology Localization and Temporal Feature Aggregation. *arXiv preprint arXiv:1807.01438*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going Deeper With Convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826.

Tang, S., Andriluka, M., and Schiele, B. (2014). Detection and Tracking of Occluded People. *International Journal of Computer Vision*, 110(1):58–69.

Tian, Y., Luo, P., Wang, X., and Tang, X. (2015). Deep Learning Strong Parts for Pedestrian Detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1904–1912.

Tomé, D., Bondi, L., Baroffio, L., Tubaro, S., Plebani, E., and Pau, D. (2016). Reduced Memory Region Based Deep Convolutional Neural Network Detection. In *2016 IEEE 6th International Conference on Consumer Electronics-Berlin (ICCE-Berlin)*, pages 15–19. IEEE.

Tomè, D., Monti, F., Baroffio, L., Bondi, L., Tagliasacchi, M., and Tubaro, S. (2016). Deep Convolutional Neural Networks for Pedestrian Detection. *Signal processing: image communication*, 47:482–489.

Valiati, G. and Menotti, D. (2018). A Preliminary Evaluation of Pedestrian Detection on Real-World Video Surveillance. In *2018 International Conference on Image Processing, Computer Vision, & Pattern Recognition (IPCV)*.

Varga, D. and Szirányi, T. (2017). Robust real-time pedestrian detection in surveillance videos. *Journal of Ambient Intelligence and Humanized Computing*, 8(1):79–85.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE.

Viola, P., Jones, M. J., and Snow, D. (2003). Detecting pedestrians using patterns of motion and appearance. *Ninth IEEE International Conference on Computer Vision*.

Wang, W., Chen, X., Zhang, G., Qian, J., Wei, P., Wu, B., and Zheng, H. (2018). Precision Security: Integrating Video Surveillance with Surrounding Environment Changes. *Complexity*, 2018.

Wojek, C., Walk, S., and Schiele, B. (2009). Multi-cue onboard pedestrian detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 794–801. IEEE.

Xing, W., Bai, P., Zhang, S., and Bao, P. (2017). Scene-specific pedestrian detection based on transfer learning and saliency detection for video surveillance. *Automatic Control and Computer Sciences*, 51(3):180–192.

Xingjian, S., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. (2015). Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *Advances in neural information processing systems*, pages 802–810.

Ye, Q., Zhang, T., Qiu, Q., Zhang, B., Chen, J., and Sapiro, G. (2016). Self-learning scene-specific pedestrian detectors using a progressive latent model. *CoRR, abs/1611.07544*, 2.

Yilmaz, E. and Aslam, J. A. (2006). Estimating average precision with incomplete and imperfect judgments. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 102–111. ACM.

Zhang, L., Lin, L., Liang, X., and He, K. (2016a). Is Faster R-CNN Doing Well for Pedestrian Detection? In *European Conference on Computer Vision*, pages 443–457. Springer.

Zhang, S., Benenson, R., Omran, M., Hosang, J., and Schiele, B. (2016b). How Far are We from Solving Pedestrian Detection? In *CVPR*.

Zhang, S., Benenson, R., and Schiele, B. (2015). Filtered Channel Features for Pedestrian Detection. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhang, S., Benenson, R., and Schiele, B. (2017). Citypersons: A diverse dataset for pedestrian detection. *arXiv preprint arXiv:1702.05693*.

Zhao, Z.-Q., Zheng, P., tao Xu, S., and Wu, X. (2018). Object Detection with Deep Learning: A Review. *IEEE transactions on neural networks and learning systems*.

Zheng, L., Yang, Y., and Hauptmann, A. G. (2016a). Person Re-identification: Past, Present and Future. *CoRR*, abs/1610.02984.

Zheng, L., Zhang, H., Sun, S., Chandraker, M., and Tian, Q. (2016b). Person re-identification in the wild. *arXiv preprint arXiv:1604.02531*.